NASA Contractor Report 166122 (Revised)

Correction, Improvement and Model Verification of CARE III, Version 3

D. M. Rose, J. W. Manke,
R. E. Altschul and D. L. Nelson

BOEING COMPUTER SERVICES
SEATTLE, WASHINGTON 98124

Contract NAS1-16900
December 1987

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

1. SUMMARY

CARE III is a reliability program designed for the assessment of fault-tolerant flight control systems. This program was developed by Raytheon under the direction of Dr. J. J. Stiffler (NASA CR-3566). CARE III, Version 3, the most recent Raytheon developed version of CARE III, was the version of the code used for this study.

Under NASA funding and direction, BCS was to verify the mathematical model and code (Task 1) and test stress the program (Task 2).

During this study, several problems with CARE III were identified. These problems concerned:

- Mathematical Modeling
- Numerical Procedures
- Code Implementation
- Use as a Design Tool

A subset of these problems was identified which could be readily addressed. A number of code modifications (Tasks 3 and 4) are described in this document. The resulting code, delivered by BCS to NASA in February 1984, is referred to as CARE III, Version 4. The problems addressed under Tasks 3 and 4 were:

- **MARKOV COVERAGE**

  The coverage module in Version 3 was numerically unstable. For the special case of a Markov coverage model, one with constant transition rates, a numerically stable solution was implemented in Version 4 which is also highly efficient. This solution is described in Section 2.0.

- **SYSTEM FAULT TREE**

  System failure due to spares exhaustion is represented in CARE III by a system fault tree. As implemented in Version 3, the calculation of

system unreliability does not completely represent the system fault tree. In particular, the contribution of coverage failure to the system unreliability may be neglected for some significant cases. The improved fault vector selection procedure for Version 4 is described in Section 3.

- **SUBRUNS**

CARE III has size limitations on the critical pair fault trees (70 modules, 20 stages). To permit the handling of larger problems, the system may be broken up into SUBRUNS, which are combined for system assessment. As implemented in Version 3, the calculation of system unreliability from SUBRUN unreliability does not assure a conservative estimate of system unreliability. This problem is discussed in Section 4.1. An improved heuristic for extracting SUBRUN fault trees from the system fault tree for Version 4 is described in Section 4.2. Also, an improved fault vector generator was developed which improves the run time for large problems.

- **MATHEMATICAL MODEL AND IMPLEMENTATION**

The mathematical model implemented in CARE III was verified for non-transient faults (CR-166096). Under Task 4, it was also verified for transient faults. The code has been modified in Version 4 to implement the model correctly for transient faults. The implementation of the sparing rules has also been corrected. Additional code changes were also made to improve the computational efficiency. A discussion of these efforts is given in Section 5.0.

- **TEST STRESSING**

As part of the assessment of CARE III as a reliability tool, two real fault-tolerant flight control systems were examined. Although FTMP is a complex system with complications that are not easily represented, CARE III offers sufficient flexibility to permit a realistic reliability evaluation. Although SIFT is very simple in design, it is

not amenable to analysis with CARE III. This is because SIFT is composed of an active pentaplex with spares. CARE III is designed to handle only duplex monitoring and triplex voting for fault tolerance. Section 6.0 provides a description of the analyses performed.

2. THE MARKOV COVERAGE MODEL

The Coverage models characterize the system handling of faults. The Single Fault Coverage model, SFCM, describes failures due to lack of fault detection in a single module. The Double Fault Coverage model, DFCM, describes failures due to coexisting faults on critical pairs of modules. Both models, presented in NASA CR-3566 and NASA CR-166096, are defined as semi-Markov processes with exponential and/or uniform transition distributions.

A special case arises when all transitions occur according to constant rates, i.e., exponential transition distributions. The coverage models then become homogeneous Markov processes. The structure of these processes allows for a larger choice of solution techniques than those proper for Semi-Markov models.

The following section describes the general approach for solving a time-homogeneous Markov process. This framework is referred to in Sections 2.2 and 2.3 in the solution of the Markov coverage models SFCM and DFCM.

2.1 MARKOV PROCESSES

A Markov process is the probabilistic model that describes the dynamics of a memory-less system, i.e., a system where the future behavior is independent of the past when the present state is known.

In such processes, transitions between states occur at constant rates and the probabilistic behavior is given by a system of ordinary differential equations.

In the Coverage models there are a finite number of states which will be numbered consecutively; state 1 is the initial state, i.e., state A in the SFCM and state B_1A_2 in the DFCM.

The coverage functions to be computed are some state probabilities and some intensities of entry into absorbing states. The problem reduces to finding the former since the latter are linear combinations of these.

A general algorithm used to evaluate the coverage functions is composed of two steps:

1. Evaluate P(t), the vector of state probabilities for non-absorbing states.

   P(t) is obtained as the solution to the system of ordinary differential equations

$$\frac{d}{dt} P(t) = G_1 P(t),$$

$$P_i(0) = \begin{cases} 1 & \text{if } i \text{ is the initial state,} \\ 0 & \text{otherwise,} \end{cases}$$

   where G_1 is the transpose of the matrix of transition rates between non-absorbing states.

2. Evaluate p(t), the vector of intensities of entry into absorbing states.

   p(t) is obtained as a linear combination of P(t).

$$p(t) = G_2 P(t)$$

   where G_2 is the transpose of the matrix of rates for transition from non-absorbing to absorbing states.

In the next two sections this algorithm is adapted to the characteristics of the two coverage models and to the specific functions to be evaluated in each case.

2.2 MARKOVIAN SINGLE FAULT COVERAGE MODEL

Under the assumption that all transitions occur at constant rates, the SFCM becomes a Markov process with states, transitions, and rates as shown in Figure 2.2-1.

The functions required by the Macro Reliability Model as outputs from the SFCM are

$p_F(t)$: intensity of entry into failure state F,
$p_{DP}(t)$: intensity of entry in detected as permanent state DP,
$P_B(t)$: probability of benign state B,
$P_{\bar{B}}(t)$: probability of non-benign state $\bar{B}$, and
$P_L(t)$: probability of latent state L,

where B = aggregate of states A, A_E and B_E;

and L = B for transient faults,
aggregate of states B and $\bar{B}$ otherwise.

The desired functions are obtained as follows:

(i) Compute the state probabilities for the states A, B, A_E and B_E ($P_i(t)$, i=1,2,3,4) by solving the four dimensional system of differential equations.

$$\frac{d}{dt} P(t) = G\,P(t),$$

where G is given by

$$\begin{bmatrix} -(\alpha+\delta P_A+p) & \beta & (1-P_A)\varepsilon c & 0 \\ \alpha & -\beta & 0 & (1-P_B)\varepsilon c \\ p & 0 & -(\varepsilon+\alpha) & \beta \\ 0 & 0 & \alpha & -(\varepsilon+\beta) \end{bmatrix}$$

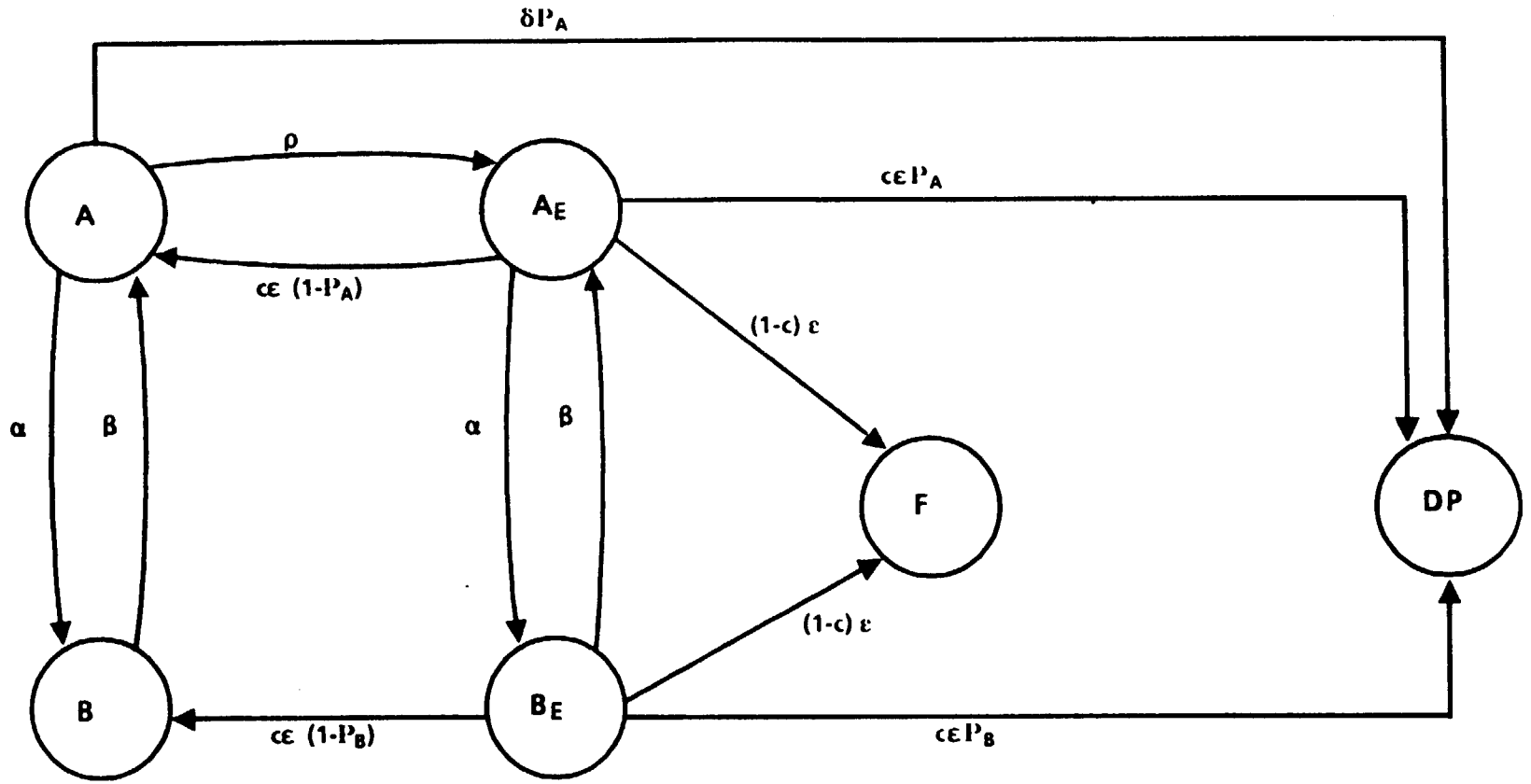


Figure 2.2-1 Markovian Single Fault Coverage Model

(ii) Evaluate the required functions as linear combinations of the functions obtained in (i). The specific calculations for each function and each fault type are shown in Table 2.2-1.

2.3 MARKOVIAN DOUBLE FAULT COVERAGE MODEL

The Markovian DFCM is shown in Figure 2.3-1.

The only function required as output from the DFCM is $p_{DF}(t)$: intensity of entry into the failure state DF.

This function is evaluated as

$$p_{DF}(t) = \lambda_2 P_1(t) + \lambda_1 P_2(t)$$

where the vector $P(t) = \left(P_1(t), P_2(t), P_3(t)\right)$ is the solution to the system

$$\frac{d}{dt} P(t) = G\, P(t),$$

with matrix G given by

$$\begin{bmatrix} -(\beta_1+\gamma_2) & 0 & \beta_2 \\ 0 & -(\beta_2+\gamma_1) & \beta_1 \\ \alpha_2 & \alpha_1 & -(\beta_1+\beta_2) \end{bmatrix}$$

2.4 IMPLEMENTATION OF MODEL

As shown in the previous section, the coverage model may be formulated as a system of ordinary differential equations (ODE's) for the Markov case. The single fault model is fourth order and the double fault model is third order. Solution of the Markov model as a system of ODE's, rather than as a system of Volterra integral equations, has several advantages. Software for the numerical solution of ODE's is available that provides high order, variable stepsize and numerically stable solutions. These features may be combined to develop a reliable solution procedure for the Markov case that is highly

TABLE 2.2-1 OUTPUT FUNCTIONS FROM SFCM

OUTPUT FUNCTIONS	FAULT TYPES		
	PERMANENT $\alpha = \beta = 0$	INTERMITTENT $\alpha, \beta > 0$	TRANSIENT $\alpha > 0.\ \beta = 0$
$P_B(t)$	----- (*)	$P_2(t)$	-----
$P_B(t)$	$P_1(t) + P_3(t)$	$P_1(t) + P_3(t) + P_4(t)$	$P_1(t) + P_3(t) + P_4(t)$
$P_L(t)$	$P_1(t) + P_3(t)$	$P_1(t) + P_2(t) + P_3(t) + P_4(t)$	$P_1(t) + P_3(t) + P_4(t)$
$\rho_{DP}(t)$	-----	-----	$\delta P_A\ P_1(t) + c\varepsilon P_A P_3(t) + c\varepsilon P_B\ P_4(t)$
$\rho_F(t)$	$(1-c)\ \varepsilon\ P_3(t)$	$(1-c)\ \varepsilon\ [P_3(t) + P_4(t)]$	$(1-c)\ \varepsilon\ [P_3(t) + P_4(t)]$

(*) These functions are not required.

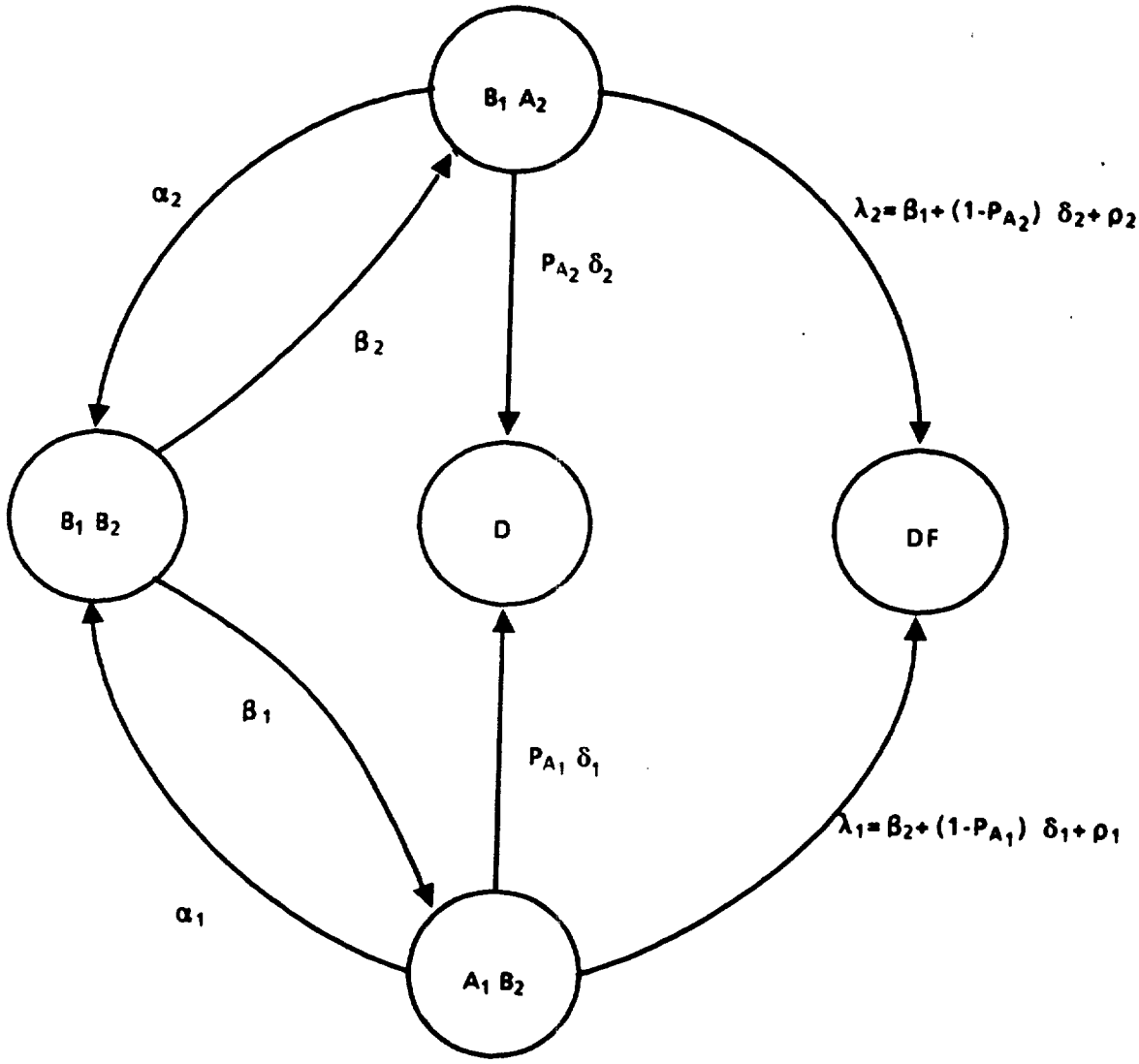


Figure 2.3-1 Markovian Double Fault Coverage Model

accurate, yet efficient. In Task 3, BCS implemented the GEARB algorithm for ODE's in Version 4; it has proven to be efficient (up to 200 times faster than the Version 3 code for solving the same Markov model) and numerically stable.

Implementation of the ODE solution method for the Markov coverage model required the addition of eight new subroutines to the COVRGE module and inclusion of the GEARB numerical integration package (HSGEAR). The Version 4 code provides the user the option to use the Version 3 solution procedure or the Version 4 method for the Markov case. (Variable MARKOV in NAMELIST set FLTTYP may be set to 1 (default) to select the Version 4 method). Figures A.1-2 to A.1-4 illustrate the structure of the Version 3 and Version 4 code and show which modules were modified or added.

For the single fault coverage model, subroutine MSNGFN computes the solution using HSGEAR. Subroutine MSNGFD is used by HSGEAR to evaluate the derivatives of the state probabilities. After the coverage model is solved, the moments of the output coverage functions are evaluated by MSNGMT using HSGEAR. Subroutine MSNGMD is used by HSGEAR to evaluate the integrand for the moment calculation. For the double fault coverage model, a procedure similar to the single fault case is used to compute the solution and moments of the output coverage functions using subroutines MDBLFN, MDBLFD, MDBLMT and MDBLMD.

3. SYSTEM FAULT TREE ANALYSIS

In the CARE III program the system unreliability is computed by the equation:

$$1 - R(t) = \sum_{\underline{\ell} \in L} Q(t|\underline{\ell}) + \sum_{\underline{\ell} \in \bar{L}} P^*(t|\underline{\ell}),$$

where $\bar{L}$ is the set of fault vectors $\underline{\ell}$ for which the system has failed due to spares exhaustion as defined by the system fault tree. Fault vectors are generated in sets by subroutine GNFLTVC in the CARE3 module. For each fault vector, logic in GNFLTVC determines whether $Q(t|\underline{\ell})$ or $P^*(t|\underline{\ell})$ is computed and summed into the unreliability. For the case of no user supplied system fault tree, $Q(t|\underline{\ell})$ is computed for any $\underline{\ell}$ for which no stage is failed by exhaustion; otherwise $P^*(t|\underline{\ell})$ is computed. This logic is consistent with the assumption that the default system fault tree is an OR tree, i.e., the system fails if any stage fails by exhaustion. For the case of a user supplied system fault tree, $Q(t|\underline{\ell})$ is computed only for those $\underline{\ell}$ selected by GNFLTVC; $P^*(t|\underline{\ell})$ is not computed for any $\underline{\ell}$. In this case, the sum of P* is computed in CARE3 directly from the minterm file for the system fault tree generated by FTREE.

Several problems with the GNFLTVC fault vector selection and generation procedure for the sum of Q calculation where identified in Tasks 1 and 2:

- $Q(t|\underline{\ell})$ is not computed for all $\underline{\ell} \in L$,
- $Q(t|\underline{\ell})$ is computed for some $\underline{\ell} \in \bar{L}$,
- Inefficient $\underline{\ell}$ generation algorithm.

Review of the GNFLTVC code and test runs indicated that $Q(t|\underline{\ell})$ may not be computed for some $\underline{\ell}$ for which the value of $Q(t|\underline{\ell})$ is a significant term in the sum of Q calculation. In addition the user had no control over the selection procedure. The algorithm for generating fault vectors in GNFLTVC generates all fault vectors, although $Q(t|\underline{\ell})$ may be computed for only a small number of vectors. The fault vector selection procedure was corrected with the Task 3 modifications and the generation algorithm was improved with the Task 4 changes.

3.1 FAULT VECTOR PROCESSING

In order to assure that all $\underline{\ell} \in L$ are processed and that the user may control which $Q(t|\underline{\ell})$ are ignored as insignificant, two capabilities are required:

- ability to test whether or not a given $\underline{\ell} \in L$
- ability to determine when $Q(t|\underline{\ell})$ is small.

The system minterm file generated by FTREE can be used to address the first requirement. Let the vector

$$\underline{\tau} = \{\tau(x) \ : \ x=1,2,\ldots,NSTGES\},$$

where $\tau(x) = 0$ or 1 be a system minterm; then a fault vector $\underline{\ell} \in \overline{L}$ if $\ell(x)>n(x)-m(x)$ for all x for which $\tau(x) = 1$, i.e., $\underline{\ell}$ "covers" $\underline{\tau}$. Thus $\underline{\ell} \in L$ only if $\underline{\ell}$ does not cover any minterm in the system minterm file. Implementation of this test requires that the system minterms be stored in core in a data structure designed to test efficiently whether a given fault vector covers any minterm.

The second requirement can be addressed by choosing a different partition of the fault vectors into sets for GNFLTVC. Let the sets L_n be defined as follows:

$$L_n = \{\underline{\ell} : 0 \le \ell(x) \le n(x), \ \sum_{x=1}^{N} \ell(x) = n\}, \ n = 0,1,2,\ldots N_{MAX}$$

where

N = number of stages in the system,

$$N_{MAX} = \sum_{x=1}^{N} n(x).$$

The L_n cover L in the sense that:

$$L = \bigcup_{n=0}^{N_{MAX}} (L_n \cap L).$$

In addition the values of $Q(t|\underline{\ell})$ are decreasing over the L_n in the sense that the numbers:

$$Q_n = max\{Q(t|\underline{\ell}) \mid \underline{\ell} \in L_n \cap L\}$$

are monotonically decreasing for $n \geq 2$.

Thus, if GNFLTVC is modified to generate fault vectors in the sets L_n, n=0,1, .., N_{max}, it is possible to be sure that $Q(t|\underline{\ell})$ is computed for all $\underline{\ell} \in L$. In addition it is possible to identify an n_0 for which $Q(t|\underline{\ell})$ is less than a user specified tolerance for all $\underline{\ell} \in L_n$ where $n \geq n_0$. Furthermore, the fault vectors in L_n may be generated by a simple algorithm that does not generate any vectors outside L_n.

3.2 IMPLEMENTATION OF FAULT VECTOR PROCEDURE

Version 3 of the CARE III program was modified to implement the fault vector selection procedure discussed in Section 3.2. The modified code, Version 4, provides the user the option to use the Version 3 selection procedure or the Version 4 selection procedure. (Variable IVSN in the NAMELIST set RNTIME may be set to 3 or 4 (default).) As illustrated in Figure A.1-5, the unreliabiity for a SUBRUN is computed by subroutine RLSBRN in the CARE3 module. If the Version 3 selection procedure is requested, RLSBRN calls NFLTVDP and GNFLTVC just as in the Version 3 code. If the Version 4 selection procedure is requested, RLSBRN calls NFLTVDP, then RDSPS to load the system minterm data into core, and finally GNFLTS to compute the SUBRUN unreliability.

Subroutine GNFLTS generates fault vectors in the sets L_n defined in Section 3.1, calls subroutine PRFLTS to compute $Q(t|\underline{\ell})$ or $P^*(t|\underline{\ell})$ for a fault vector and monitors the change in size of the sum of Q and sum of P* over L_n; see Figure A.1-9. The improved fault vector generation algorithm is coded directly into subroutine GNFLTS. The processing of fault vectors is terminated after set L_n if the change in the sum of Q for L_n is small compared to the size of the sum of Q and the change in the sum of P* for L_n is small compared to the size of the sum of P*. The logic for terminating the generation of fault vectors is applied for set L_n only for $n>2$ and if the user defined parameter LC did not affect the calculation of $Q(t|\underline{\ell})$ for any $\underline{\ell} \in L_n$. (Parameter QPTRNC in NAMELIST set RNTIME is used to control the termination of fault vector processing.)

Subroutine PRFLTS determines whether $\underline{\ell} \in L$ or $\underline{\ell} \in \bar{L}$ by calling subroutine CKSPS which checks to see if $\underline{\ell}$ covers any system fault tree minterm. The minterm data was processed by RDSPS and stored in a data structure in arrays ITRM and JTRM designed for efficient checking to determine if a fault vector covers some minterm. If $\underline{\ell} \in L$, PRFLTS calls UNRELQ to compute $Q(t|\underline{\ell})$, and if $\underline{\ell} \in \bar{L}$ PRFLTS calls FPSTAR to compute $P^*(t|\underline{\ell})$.

4. SUBRUN ANALYSIS

In the CARE III program the system may be partitioned into SUBRUN's, which consist of subsystems that are independent in the sense that modules in different subsystems are not critically coupled as defined by the critical pairs trees. For the case of no user supplied system fault tree, the system unreliability is computed by the equation:

$$1 - R(t) = \sum_{s} \left| \sum_{\underline{\ell}_s \in L_s} Q(t|\underline{\ell}_s) + \sum_{\underline{\ell}_s \in \overline{L}_s} P^*(t|\underline{\ell}_s) \right|,$$

where L_S is defined by the fault vector selection procedure implemented in subroutine GNFLTVC in the CARE3 module (see Section 3.). L_S may be interpreted as the set of fault vectors for SUBRUN-S for which no stage in SUBRUN-S is failed by exhaustion. This corresponds to the natural decomposition of the default system OR tree into an OR fault tree for each SUBRUN-S.

For the case of a user supplied system fault tree, the system unreliability is computed by the equation:

$$1 - R(t) = \sum_{s} \left| \sum_{\underline{\ell}_s \in L_s} Q(t|\underline{\ell}_s) \right| + \sum_{\underline{\ell} \in \overline{L}} P^*(t|\underline{\ell})$$

where $\overline{L}$ is the set of fault vectors $\underline{\ell}$ for which the system has failed due to spares exhaustion as defined by the system fault tree and L_S is defined by the fault vector selection procedure implemented in subroutine GNFLTVC in the CARE3 module (see Section 3.). The sum of P* is computed in CARE3 directly from the minterm file for the system fault tree generated by FTREE. Due to the problems in the Version 3 fault vector selection procedure, it is not possible to give an interpretation of L_S for this case. Furthermore, the CARE III documentation does not specify any procedure for extracting a SUBRUN fault tree from the system fault tree.

4.1 SYSTEM FAULT TREE PROCESSING

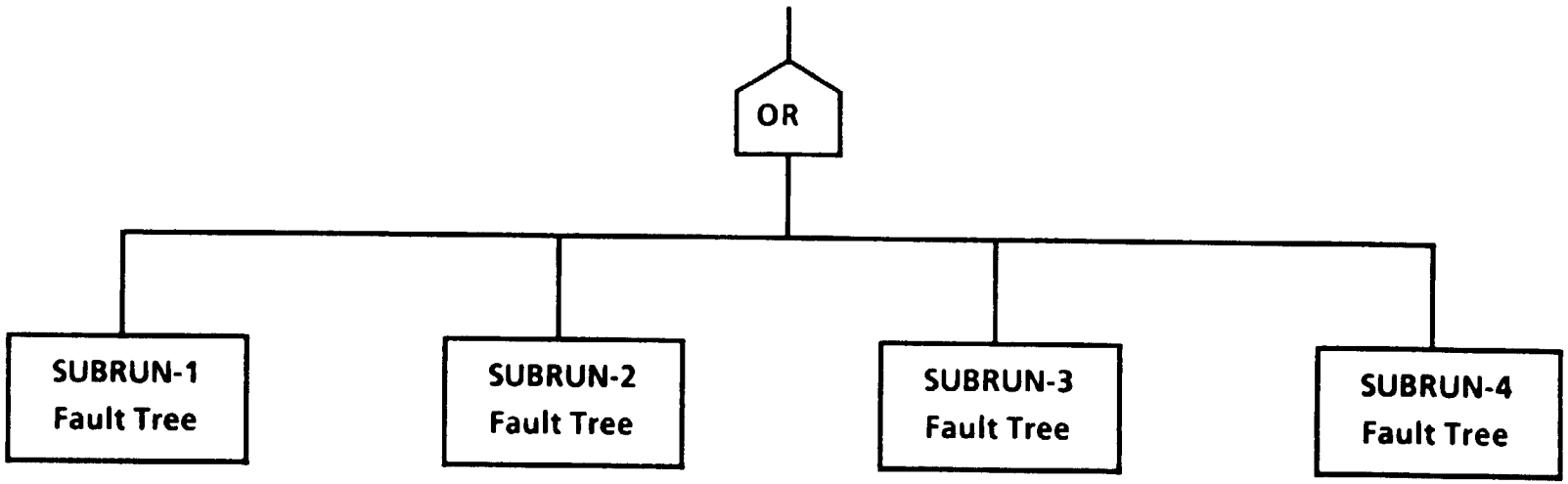
Suppose that the system fault tree is an OR with respect to the SUBRUN decomposition, i.e., the system fault tree is an OR over a set of subtrees, each of which has stages in only one SUBRUN (see Figure 4.1-1). The subtree corresponding to each SUBRUN may be used to define a fault tree for the SUBRUN, and then $\bar{L}_S$ is the set of fault vectors for SUBRUN-S for which the SUBRUN has failed as defined by the SUBRUN fault tree. Thus the system fault tree has a natural decomposition corresponding to the decomposition into SUBRUN's and the CARE III estimate of the system unreliability is conservative.

For the case of a system fault tree that is not an OR with respect to the SUBRUN decomposition, there is no natural decomposition of the system fault tree corresponding to the SUBRUN decomposition; therefore a heuristic procedure is required. One heuristic procedure is to extract from the set of minterms for the system fault tree the subset of minterms that include only stages within a SUBRUN. This subset of minterms defines a fault tree for the SUBRUN and L_S may be defined. With this construction, the system fault tree is approximated by the OR of the derived SUBRUN fault trees.

This heuristic has the advantage that for the cases:

- a single SUBRUN and any system fault tree, or
- multiple SUBRUN's with the system fault tree an OR with respect to the SUBRUN decomposition,

the natural decomposition of the system fault tree corresponding to the SUBRUN's is obtained and the CARE III estimate of the system unreliability is conservative. It has the disadvantage that in the general case, the estimate of the system unreliability may be non-conservative since some failure events are ignored.



No stages are common to any two SUBRUN fault trees

Figure 4.1-1 OR System Fault Tree

Implementation of the heuristic for extracting SUBRUN fault trees from the system fault tree requires the capability of determining when a minterm includes only stages within a SUBRUN. Let the vector

$$\underline{\tau} = \{\tau(x) : x=1,2,.\ .,\ NSTGES\}$$

($\tau(x)$ = 0 or 1) be a system minterm; then $\underline{\tau}$ includes only stages in SUBRUN-S only if

$$\sum_{x \notin SUBRUN-S} \tau(x) = 0.$$

If $\underline{\tau}$ passes this test, then the minterm for the fault tree for SUBRUN-S is defined by:

$$\underline{\tau}_S = \{\tau(x) : x \epsilon SUBRUN\text{-}S\}.$$

4.2 IMPLEMENTATION OF SYSTEM FAULT TREE PROCESSING

Version 3 of the CARE III program was modified to implement the heuristic for extracting SUBRUN fault trees from the system fault tree. The modified code, Version 4, uses the heuristic when the Version 4 fault vector selection procedure is used. For the cases:

- a single SUBRUN and any system fault tree, or
- multiple SUBRUN's with the system fault tree an OR with respect to the SUBRUN decomposition,

the Version 4 code will provide a conservative estimate of the system unreliability. For a general system tree the estimate of system unreliability for multiple SUBRUN's may be non-conservative. When the Version 3 fault selection procedure is used, the Version 4 heuristic is not applied because the Version 3 fault selection procedure does use the SUBRUN fault tree. In this case, the concerns about fault vector selection, described in Section 3, apply to each SUBRUN calculation and the estimate of system unreliability may be non-conservative for any system fault tree.

The extraction procedure described in Section 4.1 is implemented in subroutine RDSPS, which is called by subroutine RLSBRN before the call to GNFLTS; see Figure A.1-5.

5. RELIABILITY MODEL

Complete verification of the CARE III model as applied to systems with no transient faults is given in NASA CR-166096. In that analysis it is assumed that within aggregate operational states all changes are due to fast coverage transitions. Intuitively it can be argued that the dynamics within aggregate states happen instantaneously and so the Macro model becomes a non-homogeneous Markov process. More precisely, state probabilities are expressed as renewal integrals which under the above assumptions are approximated by the forward integral equations of a non-homogeneous Markov process.

The justification of the macro model for systems susceptible to transient faults requires a finer analysis since the previously used arguments do not apply.

In Section 5.1, the complications introduced by transient faults are discussed. An intermediate model is defined from which the CARE III model is justified.

5.1 JUSTIFICATION OF THE MODEL

Analysis of the coverage model shows that a module with a non-transient fault is very rapidly removed from the system (deleted from use or causes coverage failure). A transient fault may also become benign (enters B); the fault then poses no further threat and the module enters a fault free status where it becomes exposed to new faults. A module can experience consecutive transient faults until it either experiences a non-transient fault or a transient fault causes module isolation or system failure.

At the macro level, the degradation of a system is defined by the vector $\underline{\ell} = (\ell(1), \ell(2),\ldots,\ell(x),\ldots)$ where $\ell(x)$ measures the degradation in stage-x. The comparison of non-transient and transient faults suggests that behavioral differences be reflected in the definition of the vector $\underline{\ell}$. In CARE III, $\ell(x)$ is defined as the number of stage-x modules with a non-transient fault plus the number with a detected transient fault.

With this definition of the vector $\underline{\ell}$ the assumption of fast dynamics within aggregate states is no longer valid. The Macro model is not yet justified. To illustrate this, two identical systems with three modules and one fault are analyzed. The Macro models corresponding to these systems are given in Figure 5.1-1 for a non-transient fault, and in Figure 5.1-2 for a transient fault. In these figures SFCM and DFCM represent fast transitions, whereas λ represents slow transitions. Transitions due to occurrence of a new fault, slow transitions, occur only across aggregate states in the non-transient case but can occur within aggregate states in the transient case, e.g., transition from fault-free state 0 to active state A, both in macro state G(0).

An intermediate model is defined by introducing the vector $\underline{v} = (v(1), v(2),\ldots, v(x),\ldots)$, where $v(x)$ is the number of stage-x modules with latent transient faults. The states in the intermediate model are defined as aggregates of Micro model states, as a function of the operational status of the system and the parameters $\underline{\ell}$, $\underline{v}$. Similar notation to that used for the Macro model is used, e.g., $G(\underline{\ell}, \underline{v})$ denotes an operational state and $P(t|\underline{\ell},\underline{v})$ its probability.

Applying the structure of the intermediate model to the example, see Figure 5.1-3, it can be observed that only fast transitions occur within operational states. The shortcoming of the Macro model when applied to systems with transient faults is thus avoided.

5.2 MACRO MODEL RATE DERIVATION

The analysis in the last example may be extended to general systems, and it follows that the intermediate model is approximately a non-homogeneous Markov process. The probabilities for failure states in the Macro model, $Q(t|\underline{\ell})$, are obtained as sums of renewal integrals, corresponding to the contribution of each of the micro states. More concisely,

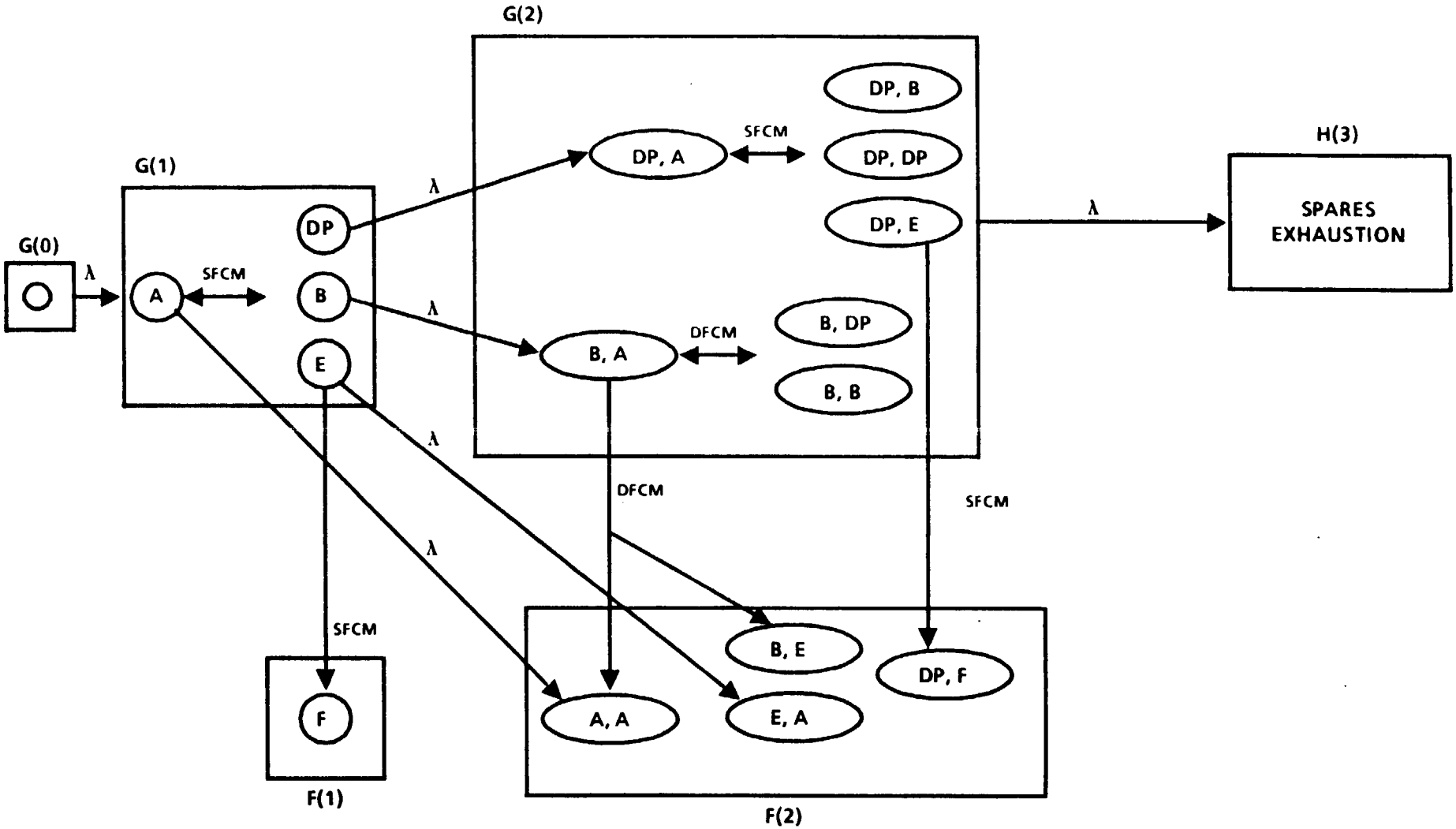


Figure 5.1-1 Macro Model - Non-Transient Fault

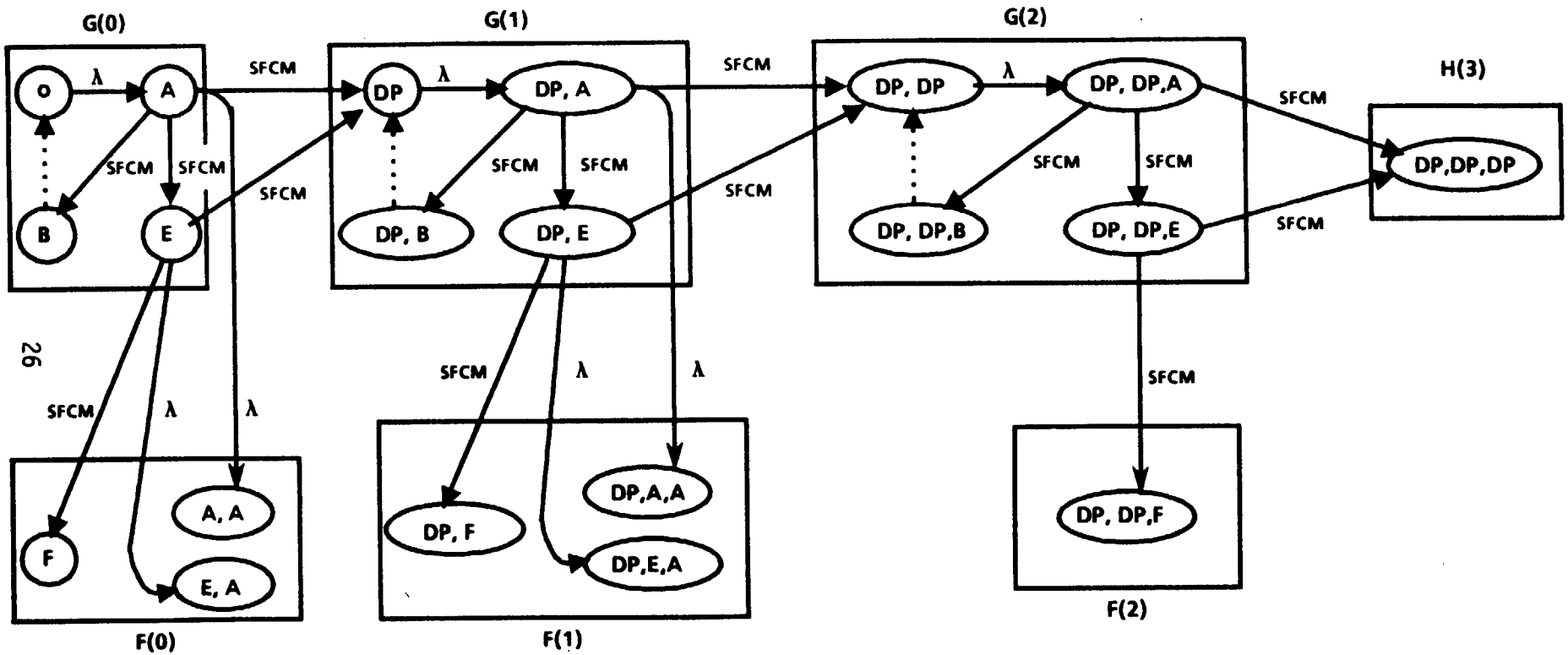


Figure 5.1-2 Macro Model - Transient Fault

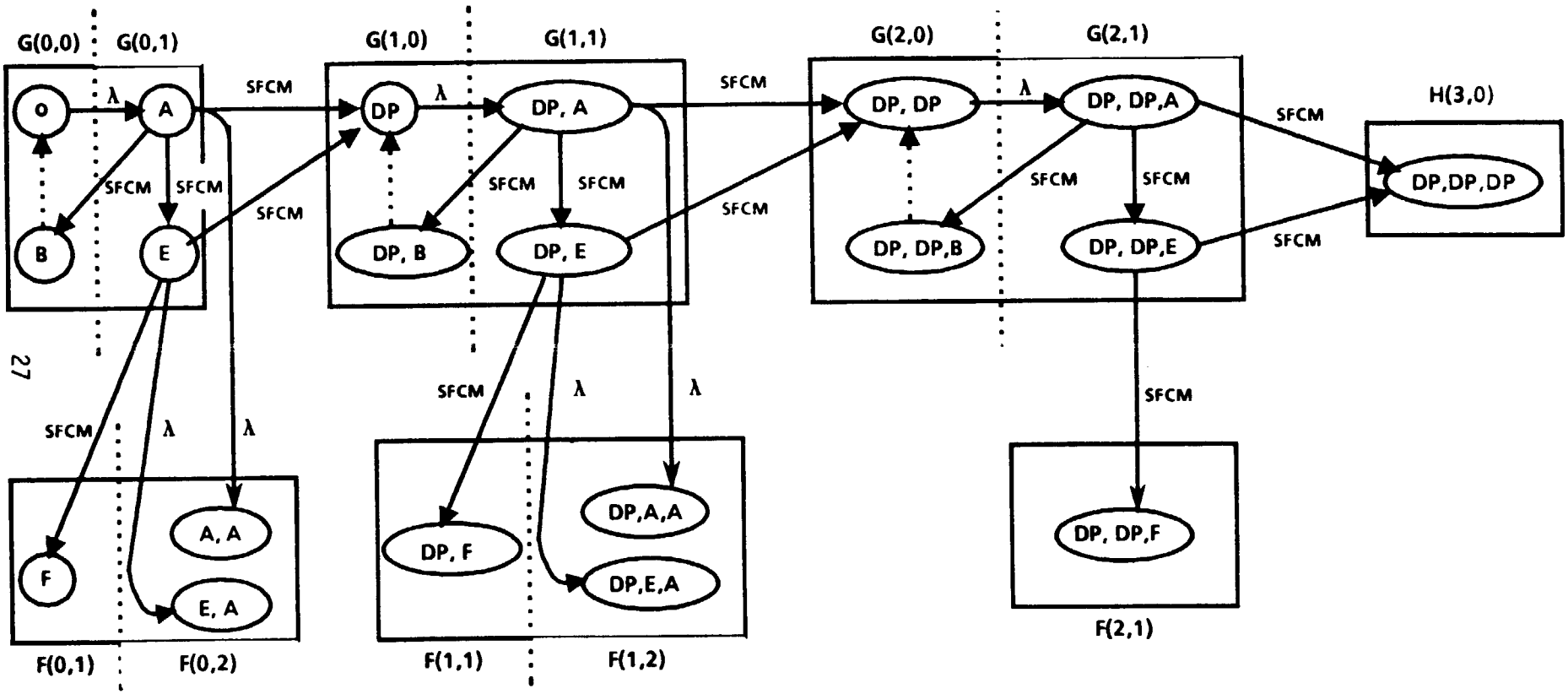


Figure 5.1-3 Intermediate Model - Transient Fault

$$Q(t|\underline{\ell}) = \int_0^\infty \left\{ P(u|\underline{\ell})\,\mu(u|\underline{\ell}) + \right.$$

$$\left. + \sum_y P(u|\underline{\ell} - \underline{1}(y))\,\lambda^{(2)}(u|\underline{\ell} - \underline{1}(y), \underline{\ell}) \right\} du \; . \qquad (5.2\text{-}1)$$

The first term in (5.2-1) corresponds to coverage failures due to latent faults or to the interaction of a new transient fault with a latent fault. The second term corresponds to double fault coverage failures due to the interaction of a new non-transient fault with a latent fault.

A conservative estimate of $Q(t|\underline{\ell})$ is obtained by allowing a larger set of risks on the operational states that lead to the failure state $F(\underline{\ell})$. This is attained by evaluating the probabilities and rates in (5.2-1) ignoring prior coverage failures. This leads to multiple counting of coverage failures and hence to conservative estimates of the reliability. Nevertheless, tight bounds are expected since fault handling occurs at several orders of magnitude faster than fault occurrence.

Under the above assumption, modules within a stage can interchange roles within an operational state. Combinatorial techniques are then possible and are used to analyze the status of modules within each stage as given in Figure 5.2-1.

The formula for $P^*(t|\underline{\ell})$, the conservative estimate for operational state probabilities, follows from simple combinatorial analyses and is given as a product of binomial probabilities. The rates are derived using the principle of inclusion and exclusion, following Riordan (1958).

The mathematical expressions of the functions used in the evaluation of coverage failure probabilities are:

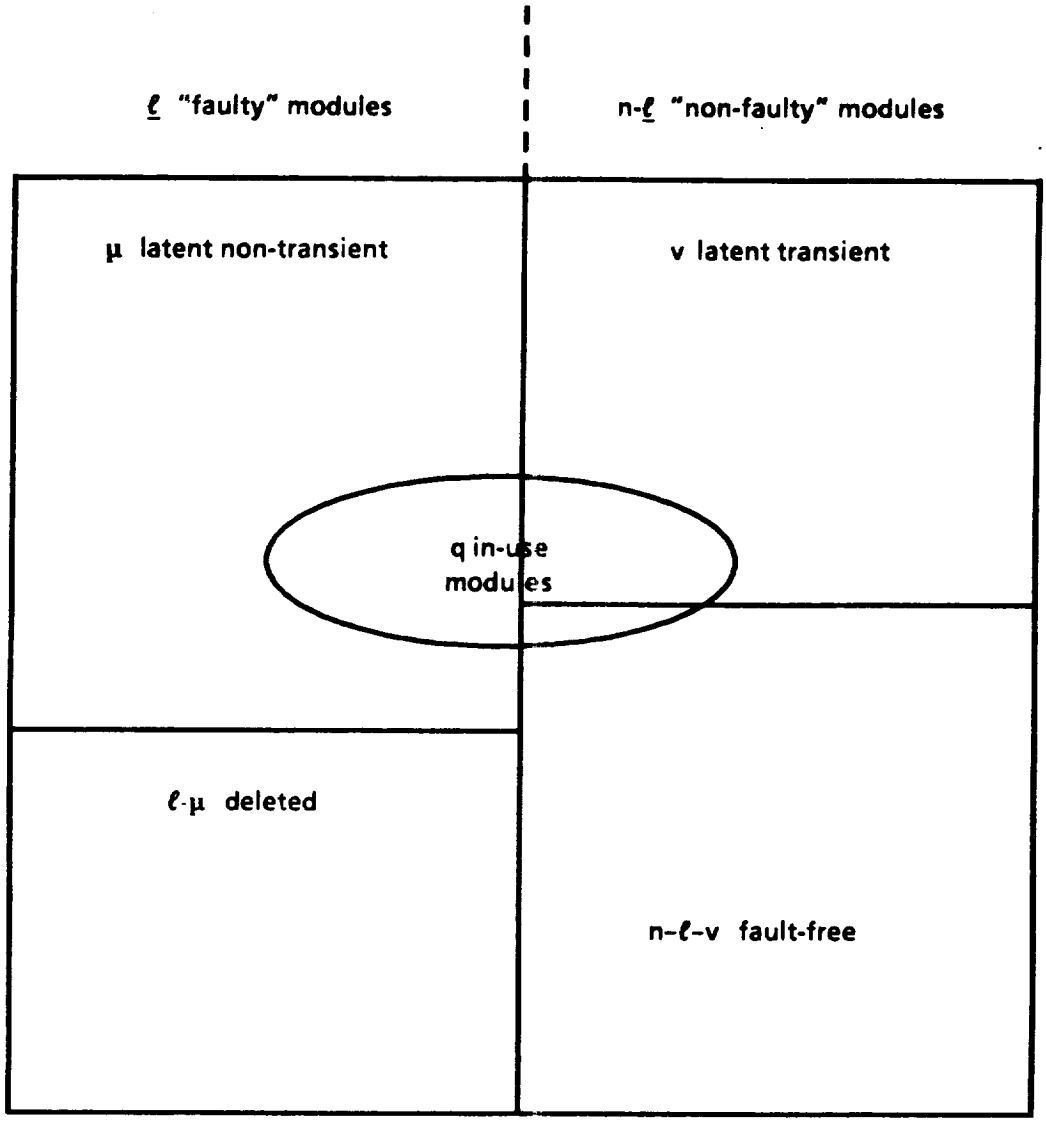


Figure 5.2-1 Status of Modules Within Each Stage

$$Q(d|\underline{\ell}) = \int_o^t K(u|\underline{\ell})\, du$$

$$K(d|\underline{\ell}) = P^*(d|\underline{\ell})\,\mu(d|\underline{\ell}) + \sum_y P^*(d|\underline{\ell} - \underline{1}(y))\lambda^{(2)}(d|\underline{\ell} - \underline{1}(y), \underline{\ell})$$

$$P^*(d|\underline{\ell}) = \prod_x \left(\frac{n(x)}{\ell(x)}\right)[1 - r(d|x)]^{\ell(x)}[r(d|x)]^{n(x) - \ell(x)}$$

$$r(d|x) = \prod_i r(d|x_i)$$

$$r(d|x_i) = \begin{cases} exp\left\{-\int_o^t \lambda(u|x_i)\, du\right\} & i \varepsilon PR(x) \\ exp\left\{-\int_o^t h_{DP}(u|x_i)\, du\right\} & i \varepsilon TR(x) \end{cases}$$

PR(x) = set of non-transient stage-x faults,

TR(x) = set of transient stage-x faults.

$$\mu(d|\underline{\ell}) = a'(d|\underline{\ell}) + A'(d|\underline{\ell}) + \lambda^*(d|\underline{\ell})$$

$$a'(d|\underline{\ell}) = \sum_x \left\{ \ell(x)\, \frac{h_F(d|x_P)}{1 - r(d|x)} + (n(x) - \ell(x))\, h_F(d|x_T) \right\}$$

$$h_F(d|x_p) = \sum_{i \varepsilon PR(x)} h_F(d|x_i)$$

$$h_F(\underline{d}x_T) = \sum_{i \in TR(x)} h_F(\underline{d}x_i)$$

$$A'(\underline{d}\underline{\ell}) = \sum_{y} \sum_{x} A'(\underline{d}\underline{\ell};(x,y))$$

$$A'(\underline{d}\underline{\ell};(x,x)) = \sum_{\mu(x)} P\left[\mu(x);\ t \mid \ell(x) \right] b_{xx}(\underline{\ell} - \underline{\mu}) \frac{\mu(x)}{H_L(\underline{d}x_p)} \cdot \left[(\mu(x) - 1) \frac{h_{DF}(\underline{d}x_p, x_p)}{H_L(\underline{d}x_p)} \right.$$

$$\left. + (n(x) - \ell(x))\, h_{DF}(\underline{d}x_p, x_T) \right]$$

$$A'(\underline{d}\underline{\ell};(x,y)) = \sum_{\mu(x)} \sum_{\mu(y)} P\left[\mu(x); \underline{d}\ell(x) \right] P\left[\mu(y); \underline{d}\ell(y) \right] b_{x,y}(\underline{\ell} - \underline{\mu}) \frac{\mu(x)}{H_L(\underline{d}x_p)}$$

$$\cdot \left[\mu(y) \frac{h_{DF}(\underline{d}x_p, y_p)}{H_L(\underline{d}y_p)} + (\mu(y) - \ell(y))\, h_{DF}(\underline{d}x_p, y_T) \right]$$

$$P\left[\mu(x);\ \underline{d}\ell(x) \right] = \binom{\ell(x)}{\mu(x)} \left[a(\underline{d}x) \right]^{\mu(x)} \left[1 - a(\underline{d}x) \right]^{\ell(x) - \mu(x)}$$

$$a(\underline{d}x) = \frac{H_L(\underline{d}x_p)}{1 - r(\underline{d}x)}$$

$$H_L(\underline{d}x_p) = \sum_{i \in PR(x)} H_L(\underline{d}x_i)$$

$$b_{x,x}(\underline{\ell} - \underline{\mu}) = \frac{N_x(q(x))}{\binom{n(x) - \ell(x) + \mu(x)}{2}}$$

$$b_{x,y}(\underline{\ell} - \underline{\mu}) = \frac{N_{x,y}(q(x), q(y))}{(n(x) - \ell(x) + \mu(x))\,(n(y) - \ell(y) + \mu(y))}$$

$$h_{DF}(\underline{d}x_p, y_p) = \sum_{i \in PR(x)} \sum_{j \in PR(y)} h_{DF}(\underline{d}x_i, y_j)$$

$$h_{DF}(t|x_p, y_T) = \sum_{i \in PR(x)} \sum_{j \in TR(y)} h_{DF}(t|x_i, y_j)$$

$$\lambda^*(t|\underline{\ell}) = \sum_x \sum_y \lambda^*(t|\underline{\ell}; (x, y))$$

$$\lambda^*(t|\underline{\ell}; (x, x)) = \lambda(t|x_T)(n(x) - \ell(x))\left[1 - H_L(t|x_T)\right] \cdot \sum_{\mu(x)=0}^{\ell(x)} P[\mu(x); t|\ell(x)]\, b_{x,x}(\ell(x) - \mu(x))\left[\mu(x)\frac{H_{\overline{B}}(t|x_p)}{H_L(t|x_p)}\right.$$

$$\left. + (n(x) - \ell(x) - 1) H_L(t|x_T)\right]$$

$$\lambda^*(t|\underline{\ell}; (x, y)) = \lambda(t|y_T)(n(y) - \ell(y))\left[1 - H_L(t|y_T)\right] \cdot \sum_{\mu(x)} \sum_{\mu(y)} P[\mu(x); t|\ell(x)]\, P[\mu(y); t|\ell(y)]$$

$$\cdot\, b_{x,y}(\ell(x) - \mu(x), \ell(y) - \mu(y)) \cdot \left[\mu(x)\frac{H_{\overline{B}}(t|x_p)}{H_L(t|x_p)} + (n(x) - \ell(x)) H_L(t|x_T)\right]$$

where

$$\lambda(t|y_T) = \sum_{j \in TR(y)} \lambda(t|y_j)$$

$$\lambda^{(2)}(t|\underline{\ell} - \underline{1}(y), \underline{\ell}) = \sum_x \lambda^{(2)}(t|x; \underline{\ell} - \underline{1}(y), \underline{\ell})$$

for x=y

$$\lambda(t|x, \underline{\ell} - \underline{1}(y), \underline{\ell}) = \lambda(t|x_p)(n(x) - \ell(x) + 1)\left[1 - H_L(t|x_T)\right] \sum_{\mu(x)=0}^{\ell(x)-1} P[\mu(x); t|\ell(x) - 1]$$

$$\cdot\, b_{x,x}(\ell(x) - \mu(x) - 1)\left[\frac{H_{\overline{B}}(t|x_p)}{H_L(t|x_p)}\mu(x) + (n(x) - \ell(x)) H_L(t|x_T)\right]$$

for $x \neq y$

$$\lambda(t|x, \underline{\ell} - \underline{1}(y), \underline{\ell}) = \lambda(t|y_p)(n(y) - \ell(y) + 1)\left[1 - H_L(t|y_T)\right] \cdot \sum_{\mu(x)} \sum_{\mu(y)} P[\mu(x); t|\ell(x)]\, P[\mu(y); t|\ell(y) - 1]$$

$$\cdot\, b_{x,y}(\ell(x) - \mu(x), \ell(y) - \mu(y) - 1) \cdot \left[\frac{H_{\overline{B}}(t|x_p)}{H_L(t|x_p)}\mu(x) + (n(x) - \ell(x)) H_L(t|x_T)\right]$$

5.3 IMPLEMENTATION OF MODEL

In tasks 3 and 4, Version 3 of the CARE III program was modified to implement the reliability model as defined in the previous section. The modified code, CARE III, Version 4, correctly implements the CARE III sparing representation defined by the NOP data and the case of transient faults. Additional code changes were made to improve the computational efficiency of the CARE3 module and to reduce the use of I/O by the code.

Implementation of the complete CARE III reliability model required modification of the input (CAREIN), coverage (COVRGE) and reliability (CARE3) modules of CARE III, Version 3. Figures A.1-1 to A.1-10 in Appendix A illustrate the structure of the Version 3 and Version 4 code and show which modules were modified or added. The overall structure of the CARE III program was not changed in the modifications. The crucial changes for the reliability model occur in subroutine CRTLPRS in module CAREIN, subroutine SNGFLT in module COVRGE and subroutines NFLTVDP, GNFLTVC and SUMMAT in module CARE3; these are discussed below.

5.3.1 Calculation of the Critical Pairs Counts (CRTLPRS)

In Version 3, the critical pairs minterm data for a SUBRUN is processed and the $b_{x,y}$ function is computed in subroutine CRTLPRS in the CAREIN module. In Version 4, the calculation of the $b_{x,y}$ function is deferred to the CARE3 module and only the critical pairs minterm data for a SUBRUN is processed in CRTLPRS (see Figure A.1-1). In Version 4, CRTLPRS is completely new and subroutines GNIQX, RDCPS and GNKXY are new code. The user's NOP data is processed by GNIQX and arrays IQXNOP and KQXNOP are established to give $q(x)$ as a function of $\ell(x)-\mu(x)$. The minterm data is read by RDCPS and critical pair counts are accumulated in array KNT by subroutine GNKXY. The KNT array contains the following data:

KNT (i(x), y, q(y)) = number of x,y critical pairs that involve module i(x) in stage-x and some stage-y module given q(y) in-use stage-y modules.

The data stored in the IQXNOP, KQXNOP and KNT arrays is sufficient for the calculation of the $b_{x,y}$ function performed in the CARE3 module. The critical pair counts $N_{x,x}(q(x))$ and $N_{x,y}(q(x),q(y))$ can be easily obtained from the KNT array.

5.3.2 Calculation of the Counts $N_{x,x}$ and $N_{x,y}$

The evaluation of the $b_{x,y}$ function requires the calculation of the counts $N_{x,x}(q(x))$ and $N_{x,y}(q(x),q(y))$. Since these counts depend only on the critical pair counts (computed in the CAREIN module), subroutine NFLTVDP in the CARE3 module was modified to call subroutine GNCPS to do the calculation (see Figure A.1-6). For each possible pair of stages x,y, GNCPS checks to see if x,y are critically coupled by checking the KNT array; x,y are critically paired only if

$$\sum_{i(x)=1}^{n(x)} KNT(i(x)\,y,\, n(y)) > 0.$$

Array IJSTGIN is used to flag whether or not x,y are critically coupled.

When stages x,y are critically paired, the counts $N_{x,x}(q(x))$ and $N_{x,y}(q(x),q(y))$ are computed using subroutines GNNXX and GNNXY and stored in arrays NXX and NXY:

$$N_{x,x}(q(x)) = \frac{1}{2} \sum_{i(x)=1}^{q(x)} KNT(i(x),x,q(x))$$

$$N_{x,y}(q(x),q(y)) = \sum_{i(x)=1}^{q(x)} KNT(i(x),y,q(y))$$

An improved version of the $b_{x,y}$ data structure and I/O scheme is used to store the NXX and NXY arrays.

5.3.3 Calculation of $b_{x,y}$ Function

As discussed in Section 5.2, the $b_{x,y}$ function depends only on $\underline{\ell}-\underline{\mu}$ and so it may be computed before the reliability model is solved. Subroutine NFLTVDP in module CARE3 was modified to call subroutine GNBPS to do the calculation (see Figure A.1-6). For each possible pair of stages x,y, GNBPS computes the $b_{x,y}$ function only if x,y are criticaly coupled as noted in the IJSTGIN array. When computation is indicated, the $b_{x,y}$ function is computed by subroutines GNBXX and GNBXY and stored in arrays BXX and BXY:

$$BXX(\ell(x)-\mu(x)) = \frac{NXX(q(x))}{N(x)-\ell(x)+\mu(x)}$$

$$BXY(\ell(x)-\mu(x),\ell(y)-\mu(y)) = \frac{NXY(q(x),q(y))}{(n(x)-\ell(x)+\mu(x))(n(y)-\ell(y)+\mu(y))}$$

The values of $\ell(x)$ and $\ell(y)$ are defined by $\underline{\ell}$ which was selected by GNFLTVC in Version 3 and GNFLTS in Version 4; q(x) and q(y) are defined by $\ell(x)-\mu(x)$ and $\ell(y)-\mu(y)$ using the IQXNOP and KQXNOP arrays; and $\mu(x)$ and $\mu(y)$ are in the range, $0\leq\mu(x)\leq\ell(x)$, $0\leq\mu(y)\leq\ell(y)$.

An improved version of the $b_{x,y}$ data structure and I/O scheme is used to store the BXX and BXY arrays.

5.3.4 Calculation of $Q(t|\underline{\ell})$

The calculation of $Q(t|\underline{\ell})$ is computed by subroutines UNRELQ and FINTGRT in the CARE3 module:

$$Q(t|\underline{\ell}) = \int_0^t K(\tau|\underline{\ell})d\tau.$$

Subroutines UNRELQ and FINTGRT were not modified in Version 4. The function $K(t|\underline{\ell})$ is computed by subroutine SUMMAT:

$$K(t|\underline{\ell}) = P^*(t|\underline{\ell})\,a'(t|\underline{\ell}) + P^*(t|\underline{\ell})A'(t|\underline{\ell}) + \sum_{y} P^*(t|\underline{\ell} - \underline{1}(y))\lambda^{(2)}(t|\underline{\ell})$$

where the first term represents single fault failures, the second term represents double fault failures with no new fault, and the third term represents double fault failures due to a new fault. In Version 3, these terms are computed in subroutines FAPC, FAC and FCYJ, respectively, and FAC and FAYJ make use of the symmetry in x,y of the $b_{x,y}$ function.

The order of calculation used in FAC and FCYJ introduces several inefficiencies into the solution of the reliability model: excessive I/O due to multiple passes through the $b_{x,y}$ data, recalculation of terms which are independent of $\underline{\ell}$ (they are only functions of time) and excessive logical tests in the inner loops of the calculation. Subroutine GNBPS in module CARE3 was modified to call subroutine GNTXX and GNTXY to evaluate all terms in the $K(t|\underline{\ell})$ calculation that depend only on time before the reliability model was solved see (Figure A.1-8). Subroutine SUMMAT and the $b_{x,y}$ data structure and I/O scheme were completely modified to eliminate the excessive use of I/O and logical tests. (If the number of pairs of critically coupled stages in the user's model does not exceed 20, then all I/O operations involving the $b_{x,y}$ data is avoided.) The Version 3 subroutines, FAC, FCYJ, FBCRTL and FDSCRTL are replaced in Version 4 by subroutines GNFXX, FB1XX, FB2XX, FB1XY and FB2XY (see Figure A.1-10).

As discussed in Section 5.1, the transient fault model introduces an extended interpretation of the fault vector $\underline{\ell}$. In Version 4, the logic in subroutines GNFLTS and SUMMAT was extended to properly include this revised interpretation of fault vectors.

5.3.5 Method of Moments

The calculation of $K(t|\underline{\ell})$ requires the evaluation of several convolution integrals:

$$y(t) = \int_0^t P_2(\tau)\, P_1(t-\tau)\, d\tau,$$

where $P_1(t)$ is a measure of the rate at which a certain class of faults occurs and $P_2(\tau)$, one of the coverage output functions, is a function of the interval τ between that occurrence and the entry of the fault into a particular coverage state. The numerical convolution procedure implemented in the CARE III module uses the method of moments. The calculation is based on two assumptions: $P_1(t)$ is a much more slowly varying function of time than $P_2(t)$; and $P_2(t)$ decays rapidly to zero. The first assumption is consistent with the CARE III assumption that coverage rates are much higher than module failure rates. However, the second assumption was not valid for the coverage output function P_{DP}. To correct this problem, subroutines SNGFLT and MSNGFN in the COVRGE module were modified to provide the intensity p_{DP} as an output instead of P_{DP}. The CARE3 module was appropriately modified to compute h_{DPT} from p_{DP} instead of P_{DP}. The overall result of these changes is a more accurate evaluation of h_{DPT}.

6. TEST STRESSING

As part of the validation task of CARE III, the reliability of hypothetical systems was evaluated. The answers obtained compared favorably with analytic results. As part of the test stressing, two fault-tolerant systems were evaluated using CARE III; Section 6.1 describes the FTMP analysis, and Section 6.2 the SIFT problem. Although FTMP presented a complex architecture for representation, CARE III offered sufficient flexibility to approximate the system. SIFT, although significantly simpler than FTMP, illustrated that CARE III is limited to simplex, duplex, and triplex systems; pentaplex (3-out-of-5) voters cannot be represented well.

6.1 FTMP

The FTMP system (NASA CR-166071,72,73) consists of ten LRU's (line replaceable units) and connecting buses. Each LRU contains a processor, a clock generator, a power supply, a memory (slave region) and two bus guardian units (BGU). For the reliability analysis, the BGU's may be lumped in with the processor; their failure rates should be added together. Similarly the memory includes the real time clock, system control register and the I/O port. There are four different types of buses: poll (P), receive (R), transmit (T) and clock (C). There are 5 of each type of bus.

Fault-tolerance is incorporated by triplex voting with majority rule (except for clocks). No single-fault coverage failures should occur. The system is initially composed of three processor triads with one spare processor, two memory triads with four memories as spares, and one clock quadruplex with six spare clocks. The modules in a triplex (quad for clocks) are rotated. At any given time the processors in a triplex may be from any of the LRU's--similarly for the memories and clocks.

The P, R and T buses each form a triplex with two spares. The C bus forms a quad with one spare. If no inter-LRU dependence existed, the minimum number of modules needed for each stage are: processor (5), memory (5), clock (3), P, R and T bus (2 each), C bus (3).

The failure rates used in the analyses are:

processor (plus 2 BGU's)	2.2×10^{-4}/hr
memory	2.0×10^{-4}/hr
clock	1.0×10^{-5}/hr
power supply	1.0×10^{-4}/hr
P, R, T and C bus	1.0×10^{-5}/hr

Dependence arises with FTMP in that if a processor within an LRU fails, no other modules are affected. If a clock or power supply fails, all the modules within the LRU may function improperly. When a clock or power supply are identified as faulty, the entire LRU is deleted from the system. A faulty slave region may not affect the operation of the rest of the LRU; however, if identified as faulty, it will cause the entire LRU to be deleted. Dependence affects the reliability of the system in two ways, in the computation of spares exhaustion failure and of coverage failure,

When assessing spares exhaustion, dependence complicates the relationship between the number of failed modules and the number of operational modules remaining. For example, if two processors fail and then two memories fail (under perfect coverage), the number of operational processors left can be six, seven or eight. This depends on whether the failed processors and memories are from the same LRU (eight processors left), different LRU's (six left), or one LRU with a processor and then a memory failure, one LRU with a processor failure and one LRU with a memory failure (seven left).

CARE III does not allow for module interdependency. An added complication is the functional numbering in CARE III, as opposed to physical numbering. Processor 1 denotes the processor that currently is performing function 1; this in FTMP would be functioning in the first triad. Processor 1 could be from any LRU. In particular, processor 1 and memory 1 will, most of the time, be from separate LRU's. A discussion of functional numbering is provided in Section 3.0 of the BCS Final Report.

As our first-cut model for spares exhaustion, all the modules within an LRU are lumped together to form a single stage. The combined stage failure rate

is the sum of the module failure rates. Perfect coverage is assumed. This model leads to a conservative evaluation (overestimate) of exhaustion failure. Figure 6.1-1 provides the control information for this FTMP model. The estimate obtained for the probability of system exhaustion failure is P*SUM = 1.45×10^{-11}.

The unreliability obtained satisfies our requirements (P significantly less than 10^{-9} for a 10 hour flight); no further analysis of exhaustion failure is required. If this conservative procedure did not provide satisfactory results, a more detailed model could be evaluated. One could let each module be a stage, and thus represent in detail what combinations of module failures cause exhaustion failure. The problem with such a representation is that the system fault tree becomes quite complex and there is an appreciable chance of user input error. This model still assumes perfect coverage since one cannot input system sparing rules and the success configuration information (NOP). For the FTMP analysis this more detailed modeling was not necessary.

Initial information on FTMP was obtained from NASA CR-166071, CR-166072 and CR-166073. Additional information and assistance on FTMP was provided by Mr. C. Liciega from NASA-Langley. The failure rate values were based on those used in the Draper reliability analyses. The coverage parameters were based on the FTMP fault injection study, CR-166073, and a description of how the system operates.

Exponential transition rates were used for the coverage analyses. The $\alpha(t)$ transition can be taken as exponential, since the transition of a faulty module from a latent state to an error generating state can be considered random in time. The detection rate, $\delta(t)$, is certainly based on how often self-tests are run. There are 37 self-test programs for the processor, clock generator and bus. A new program is run every 320 milliseconds. Once a module is detected as faulty and the error latches are set, another clocking cycle is required, 320 milliseconds, before the module can be deleted and replaced by a spare. Each test program does not detect solely a unique type of fault. Certain types of faults will be detected by many of the self-tests. A uniform distribution does not appear to describe this operation well. An exponential distribution was used, such that five percent of the

```
$FLTTYP  DEL (1) = 1.0E6,
      RHO (1) = 0.0,
      C  (1) = 1.0,
      IDELF(1)  =  1,
      IRHOF(1)  =  1,
      IEPSF(1)  =   1,
      CVPRNT    =    .TRUE.$
$STAGES  NSTGES  =  5,
      N(1) = 10,     } LRU
      M(1) = 5,      }
      N(2) = 5,      } P Bus
      M(2) = 2,      }
      N(3) = 5,      } R Bus
      M(3) = 2,      }
      N(4) = 5,      } T Bus
      M(4) = 2,      }
      N(5) = 5,      } C Bus
      M(5) = 3,      }
      IRLPCD   = 4$
$FLTCAT  RLM  ( 1,1)  =  5.3E -4, ← LRU Failure Rate
      RLM (1,2) = 1.0E -5,
      RLM(1,3) = 1.0E -5,
      RLM (1,4) = 1.0E -5,
      RLM (1,5) = 1.0E -5$
$RNTIME  FT  =  10.0,
      ITBASE  =  1,
      SYSFLG  =  .TRUE.,
    CPLFLG  =  .FALSE.$
FTMP RUN WITH LRU TREATED AS A STAGE
1 5 6 6
6 0 1 2 3 4 5
```

Figure 6.1-1 Input File for Obtaining Bound on Exhaustion Failure with Dependence

probability was to the right of 38 x .320 milliseconds. With this distribution, "more of the action" happens early, which represents the overlap in the self-test programs. Furthermore, the five percent tail to the right of the theoretical maximum detection time should lead to a slightly conservative answer.

The parameter for the exponential distribution is the inverse of the mean. Using Table 7, CR-166073, α was taken as 1/(mean time to detect error - 160 milliseconds), converted to hours. The detection time was adjusted by 160 milliseconds since, on the average, there will be that much delay between when an error is propagated and when an error flag is set.

The $\varepsilon(t)$ transition is from an active error producing state, A_E, to either a detected, identified and reconfigured state, D_p, or to coverage failure F. The length of time in the state A_E is important for assessing double-fault coverage failures; only the transition to D_p need be considered. The mean time for this transition was taken as the sum of the mean times for identification and for reconfiguration, as given in Table 7, CR-166073, plus 160 milliseconds. The 160 millisecond adjustment allows for the average time between initiation of error propagation and the setting of error flags. Using this mean time, converted to hours, provides the exponential parameter ε for the coverage model.

Coverage failure is also affected by dependence. For FTMP no single-fault failures in theory can occur; double-fault failure must address dependence. Consider three processors in a triplex (Figure 6.1-2). Any two faulty processors will defeat the majority voter; thus all pairs of processors in the triad are critically paired. But consider a clock failure in an associated LRU. This causes faulty processor operation within the LRU. Hence the clock is critically coupled to the other two processors. Similarly, the power supply and memory are critically coupled with the processors (and clocks). In order to represent this dependence, one defines an equivalence class for each LRU. A double-fault failure may occur if there is one fault in two of the equivalence classes. The control file for the LRU critical pair fault tree is given in Figure 6.1-3. The answer obtained by this representation is conservative, since some module pairs are incorrectly

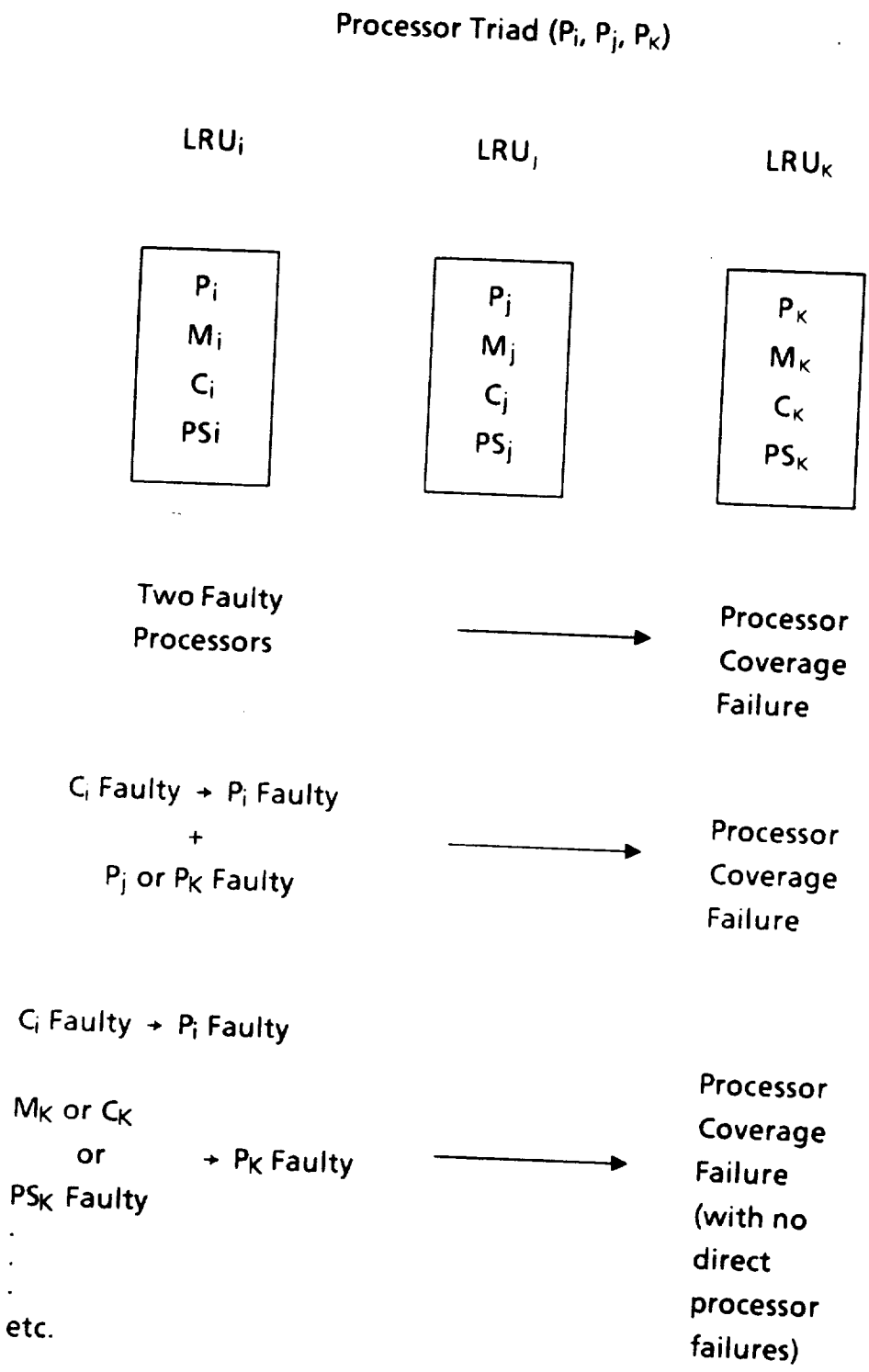


Figure 6.1-2 Dependence Effect on Coverage

```
$FLTTYP  DEL (1) = 8.87E2,
      RHO (1) = 1.91E4,
      C  (1) = 1.0,
      EPS (1) = 1.05E4,
      DEL (2) = 0.0,
      RHO (2) = 8.39E3,
      EPS (2) = 1.12E4,
      DEL (3) = 8.87E2,
      RHO (3) = 1.0E6,
      EPS (3) = 1.13E4,
      DEL (4) = 8.87E2,
      RHO (4) = 2.05E3,
      EPS (4) = 7.8E3,
      DEL (5) = 8.87E2,
      RHO (5) = 1.0E6,
      EPS (5) = 1.13E4,
      NFTYPS = 5,
      CVPRNT    =  .TRUE.$
$STAGES  NSTGES  =  8,
      N(1) = 10,    } Processor
      M(1) = 5,     }
      N(2) = 10,    } Memory
      M(2) = 5,     }
      N(3) = 10,    } Clock
      M(3) = 3,     }
      N(4) = 10,    } Power Supply
      M(4) = 5,     }
      N(5) = 5,     } P Bus
      M(5) = 3,     }
      N(6) = 5,     } R Bus
      M(6) = 2,     }
      N(7) = 5,     } T Bus
      M(7) = 2,     }
      N(8) = 5,     } C Bus
      M(8) = 2,     }
      NOP(1,1) = 9,
      NOP(2,1) = 6,
      NOP(1,2) = 6,
      NOP(1,3) = 9,
      NOP(2,3) = 6,
      NOP(1,4) = 9,
      NOP(2,4) = 6,
      NOP(1,5) = 4,
      NOP(1,6) = 3,
      NOP(1,7) = 3,
      NOP(1,8) = 3,
      IRLPCD  = 1$
```

Figure 6.1-3 Input File for Estimate of Coverage Failure, 1 Subrun

```
$FLTCAT  JTYP (1,1) = 1,
       JTYP (1,2) = 2,
       JTYP (1,3) = 3,
       JTYP (1,4) = 5,
       JTYP (1,5) = 4,
       JTYP (1,6) = 4,
       JTYP (1,7) = 4,
       JTYP (1,8) = 4,
       RLM  (1,1)  =  2.2E-4,
       RLM (1,2) = 2.0E-4,
       RLM(1,3) = 1.0E-5,
       RLM (1,4) = 1.0E-4,
       RLM (1,5) = 1.0E-5,
       RLM (1,6) = 1.0E-5,
       RLM (1,7) = 1.0E-5,
       RLM (1,8) = 1.0E-5$
$RNTIME  FT  =  10.0,
       ITBASE  =  1,
       SYSFLG  =  .TRUE.,
     CPLFLG  =  .TRUE.$
FTMP MODEL IV
1 8 9 9
9 0 1 2 3 4 5 6 7 8
CP TREE FOR MODEL IV
1 60 61 78
1 1 10
2 11 20
3 21 30
4 31 40
5 41 45
6 46 50
7 51 55
8 56 60
61 0 1 11 21 31
62 0 2 12 22 32
63 0 3 13 23 33
64 2 61 62 63
65 0 4 14 24 34
66 0 5 15 25 35
67 0 6 16 26 36
68 2 65 66 67
69 0 7 27 37
70 0 8 28 38
71 0 9 29 39
72 2 69 70 71
73 2 21 22 23 24
74 2 41 42 43
75 2 46 47 48
76 2 51 52 53
77 2 56 57 58
78 0 64 68 72 73 74 75 76 77
```

Figure 6.1-3 (Continued)

represented as being critically coupled (e.g., processor of LRU 1 and memory of LRU 2). This equivalence class representation, however, makes for an easy representation of the LRU dependence effect of processor and memory triads simultaneously. Using a single critical pair tree, the probability of a coverage failure obtained is Q SUM = 6.039×10^{-9}. Note that the upper bound for exhaustion failure, 1.45×10^{-11}, affects only the third significant digit. The problem was rerun using two critical pair trees (two subruns) with processors, memories and power supplies in the first subrun and buses in the second (Figure 6.1-4). This reduces the CPU run time and drastically reduces the amount of output. The answer obtained differed only at the seventh significant digit. In most of the detailed analyses, subruns were used, providing highly accurate results at a much lower cost. The restrictions for the use of subruns are given in Section 4.0.

6.2 SIFT

SIFT operates as a one-stage system which consists of a pentaplex of modules, plus spares. Within the pentaplex, fault tolerance is based on three out of five voting. Coexisting faults on three of the modules within the pentaplex are necessary to cause coverage failure of the system. This means that critical triplets, as opposed to critical pairs, need to be considered when assessing the probability of coverage failure.

CARE III has been suggested as an evaluation tool for such systems by disregarding failure probabilities unless enough faults are present. $Q(t|\ell)$, the probability of coverage failure when ℓ faults have occurred, is evaluated only when $\ell \geq LC$, where LC is an input parameter. For a pentaplex, LC is set equal to 3.

If transient faults are possible, then literal triplets can occur even when $\ell \leq 2$, (ℓ counts transients only when they cause the module to be isolated). The use of the LC parameter will lead in this case to very optimistic results.

If only non-transient faults are possible, the use of LC=3 gives correct results only in the case of a pentaplex with no spares. If the system

```
FTMP MODEL IV
1 8 9 9
9 0 1 2 3 4 5 6 7 8
CP TREE FOR MODEL IV - SUBRUN 1
1 40 61 74
1 1 10
2 11 20
3 21 30
4 31 40
61 0 1 11 21 31
62 0 2 12 22 32
63 0 3 13 23 33
64 2 61 62 63
65 0 4 14 24 34
66 0 5 15 25 35
67 0 6 16 26 36
68 2 65 66 67
69 0 7 27 37
70 0 8 28 38
71 0 9 29 39
72 2 69 70 71
73 2 21 22 23 24
74 0 64 68 72 73
CP TREE FOR MODEL IV - SUBRUN 2
41 60 74 78
5 41 45
6 46 50
7 51 55
8 56 60
74 2 41 42 43
75 2 46 47 48
76 2 51 52 53
77 2 56 57 58
78 0 74 75 76 77
```

Figure 6.1-4 Critical Pair Tree for Two Subruns

consists of one or more pentaplexes and spares, the reliability estimate given by CARE III is extremely conservative.

Three events contribute to $Q(t|3)$ in CARE III:

(i) three latent faults in a pentaplex;
(ii) three latent faults; two of these in a pentaplex;
(iii) two latent faults in a pentaplex, one deleted module.

Of the three cases, only the first corresponds to a true coverage failure in a pentaplex. The last two cases are included in the evaluation of $Q(t|3)$ since CARE III is based on a critical pair type architecture. The first two events are of the same order, since both cases depend on three latent faults. For the highly reliable systems being considered, fault handling is several orders of magnitude faster than fault occurrence and the third event is correspondingly considerably greater than the first two; it will be the dominating term in the evaluation of $Q(t|3)$. The corresponding coverage failure estimate will then be unacceptably conservative.

As an example, consider a system consisting of one pentaplex and one spare. Modules are susceptible to a permanent fault which occurs at constant rate λ and detection occurs at constant rate δ. If $\delta >> \lambda$ then

$$\text{TRUE UNRELIABILITY} \approx 5\left(\frac{\lambda}{\delta}\right)^2\left[1-e^{-6\lambda t}\right]$$

$$\text{CARE III UNRELIABILITY} \approx 4\left(\frac{\lambda}{\delta}\right)^2\left[1-5e^{-6\lambda t}-6e^{-5\lambda t}\right]$$

In particular if $\lambda=5\times10^{-4}$; $\delta=100$ and $t=1$, the respective values are 3.7×10^{-13} and 7.4×10^{-11}.

7. REFERENCES

Bryant, L. A. and J. J. Stiffler (1982), CARE III Phase II Report Maintenance Manual, NASA CR-165863.

Riordan, J. (1958), An Introduction to Combinatorial Analysis, John Wiley & Sons, New York.

Rose, D. M., R. E. Altschul, J. W. Manke, and D. L. Nelson (1983), Review and Verification of CARE III Mathematical Model and Code: Interim Report , NASA CR-166096.

Rose, D. M., R. E. Altschul, J. W. Manke, and D. L. Nelson (1984), Trials of the CARE III Model: Final Report (Draft), NASA Contract NAS1-16900.

Smith, T. B. and J. H. Lala (1983), The Development and Evaluation of a Fault Tolerant Multi Processor (FTMP) Computer.
Vol. 1: FTMP Principles of Operation, NASA CR-166071.
Vol. 2: FTMP Software, NASA CR-166072.
Vol. 3: FTMP Test and Evaluation, NASA CR-166073.

Stiffler, J. J., and L. A. Bryant (1982), CARE III Phase II - Mathematical Description, NASA CR-3566.

APPENDIX A

This appendix documents the modifications made to CARE III, Version 3, in Tasks 3 and 4; the modified program is referred to as CARE III, Version 4. The first section describes the changes in terms of the "call trees" of the principal modules of the program. The second section consists of the "design sheets" prepared for all the modified or new subroutines in Version 4.

A.1 CALL TREE SPECIFICATIONS

In this section an overview of the differences between the Version 3 and the Version 4 code is presented in terms of the "call trees" of the principal modules of the program. In Figures A.1-1 to A.1-10, the modified or new subroutines in Version 4 are indicated by boldface type. The figures show that the overall structure of the CARE III program was not changed in the Task 3 and 4 modifications.

A.2 DESIGN SPECIFICATIONS

The design of each of the modified or new subroutines in the Version 4 code is summarized by a "design sheet" presented in this section. These design sheets were prepared as the first step in the coding of the Version 4 changes. They are an overview of the subroutines; not all computational details are included. However, they do indicate the overall sequence of computations and the data needed for and generated by each step in the subroutine. The design sheets are presented in the following order:

- CAREIN;
  - Figures A.2-1 to A.2-5
- COVRGE: Markov model;
  - Figures A.2-6 to A.2-13
- CARE3: Main control and computation subroutines;
  - Figures A.2-14 to A.2-21

- CARE3: Calculation of NXX and NXY data;
  - Figures A.2-22 to A.2-24

- CARE3: Calculation of BXX and BXY data;
  - Figures A.2-25 to A.2-29

- CARE3: Calculation of K(t|$\underline{\ell}$);
  - Figures A.2-30 to A.2-35

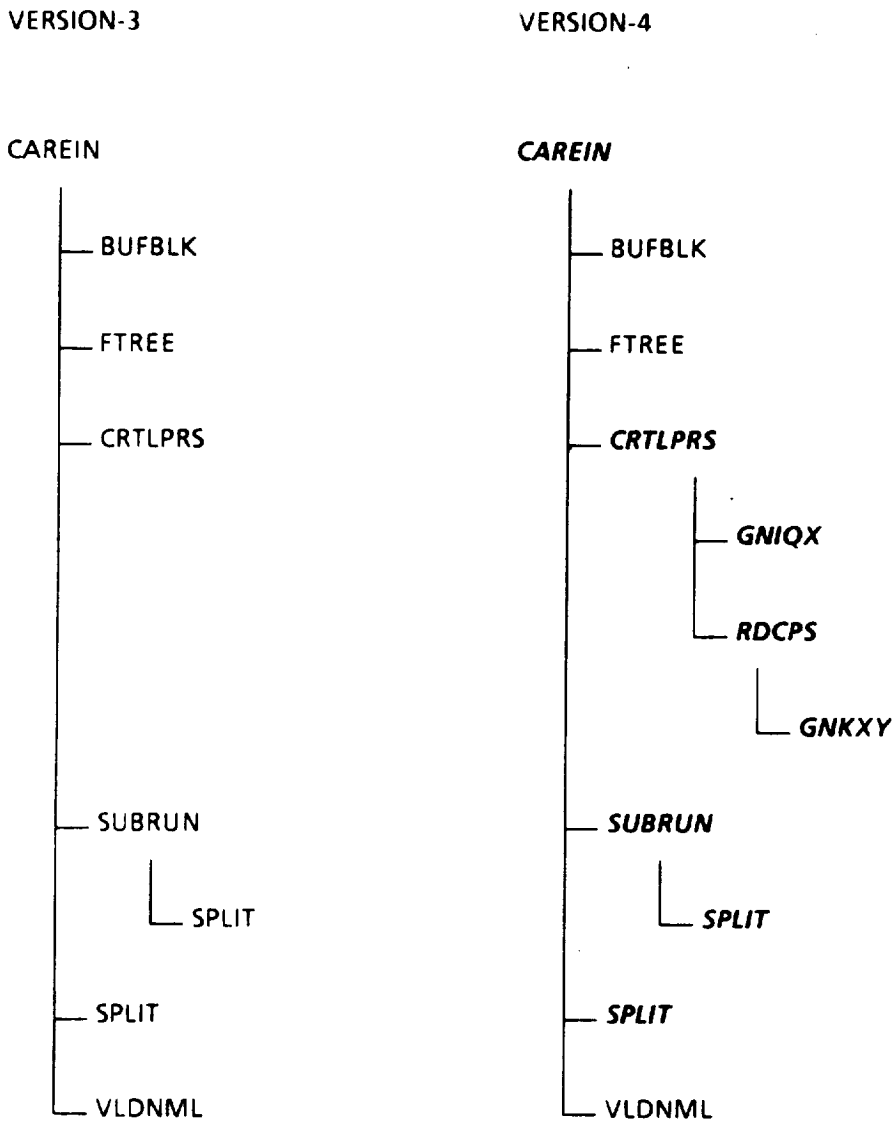


Figure A.1-1 CAREIN Call Tree

[Note: Boldface on this and following figures indicates routines that have been added or modified.]

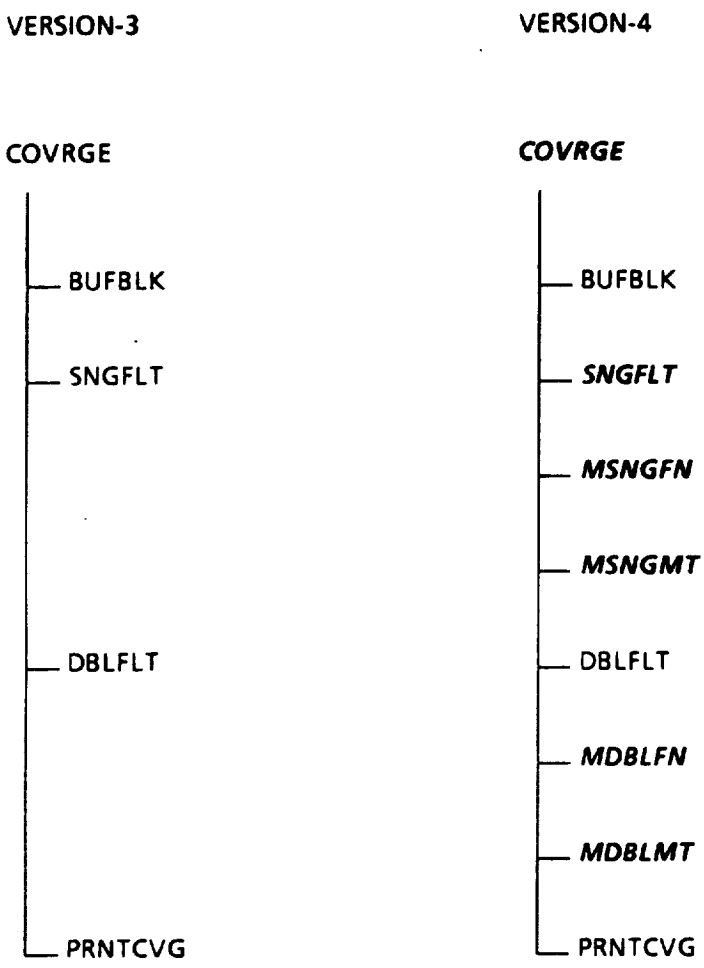


Figure A.1-2 COVRGE Call Tree

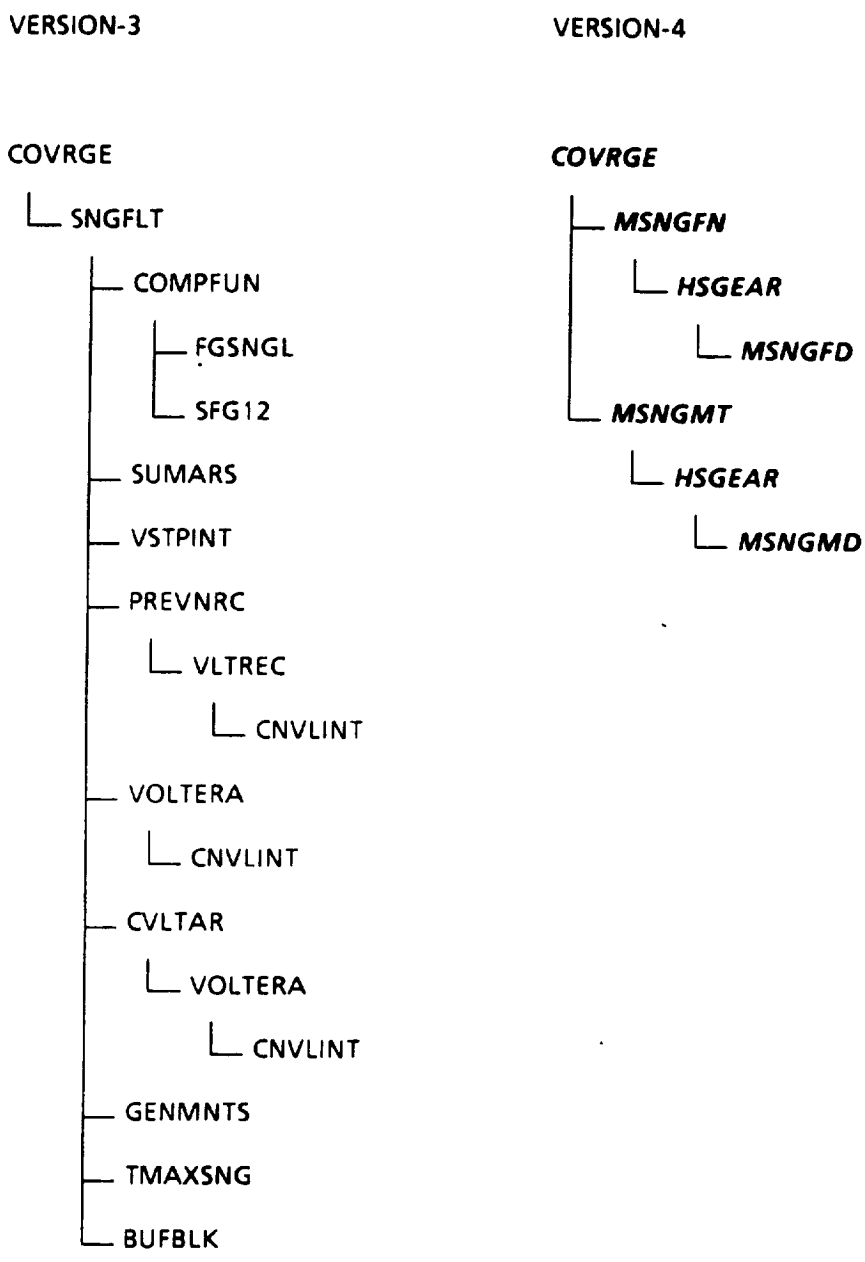
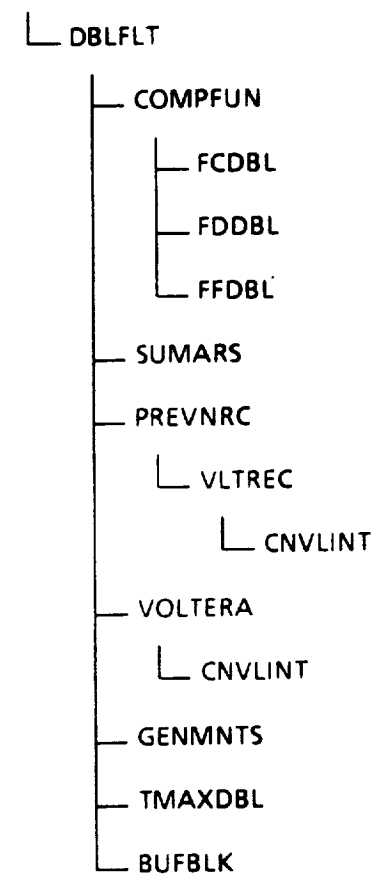


Figure A.1-3 Single Fault Call Tree

VERSION-3



VERSION-4

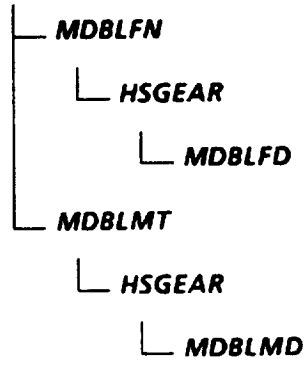


Figure A.1-4 Double Fault Call Tree

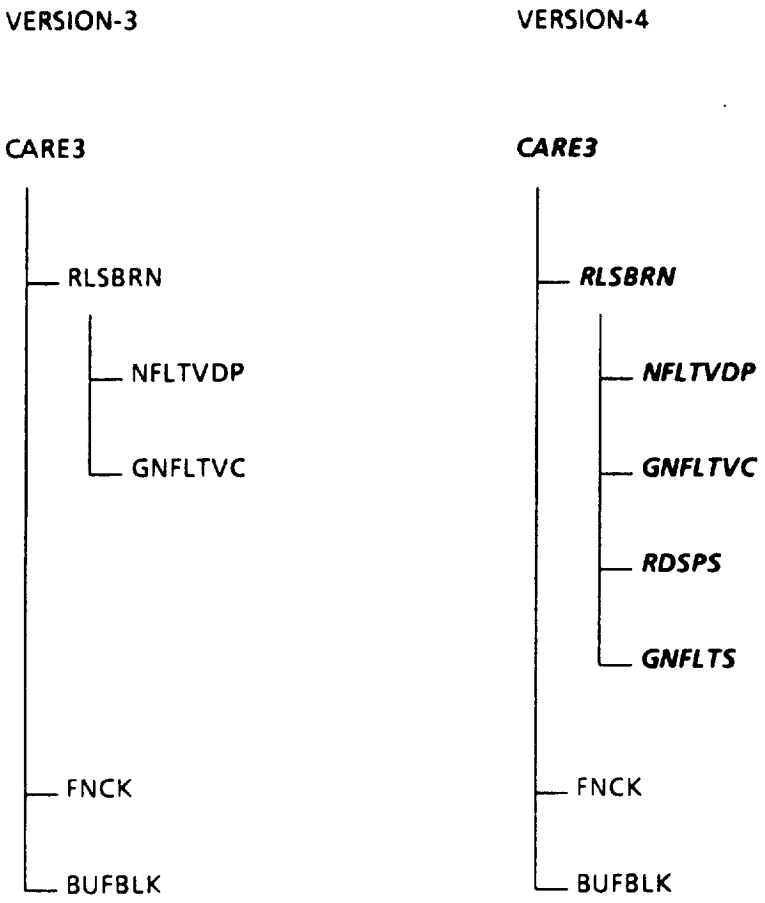
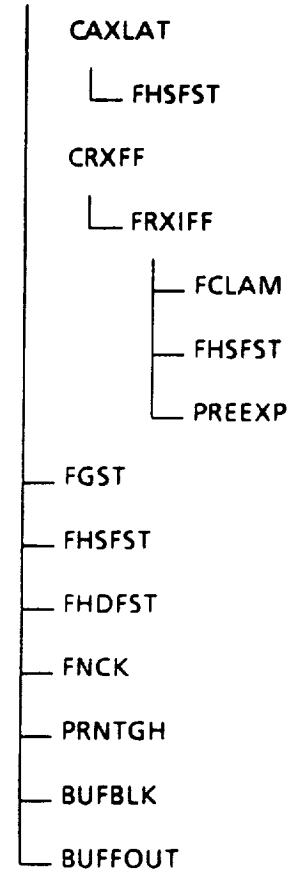


Figure A.1-5 CARE3 Call Tree

VERSION-3



VERSION-4

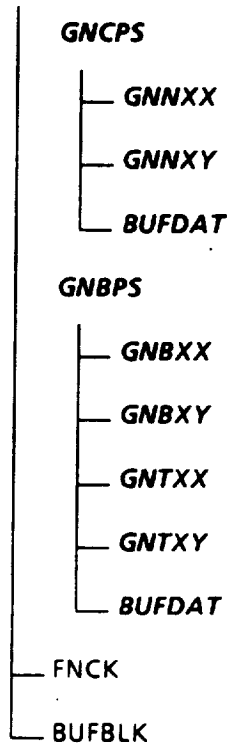


Figure A.1-6 NFLTVDP Call Tree

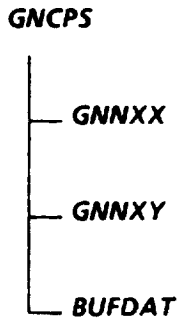


Figure A.1-7 GNCPS Call Tree

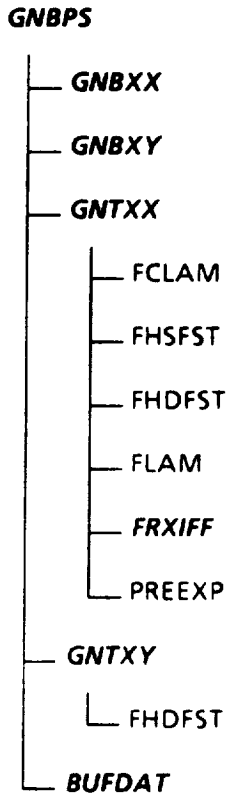


Figure A.1-8 GNBPS Call Tree

VERSION-3 FAULT VECTOR SELECTION PROCEDURE

GNFLTVC

- FPSTAR
- ARZERO
- ***GNBXY***
- UNRELQ
  - FINTGRT
  - ***SUMMAT***

VERSION-4 FAULT VECTOR SELECTION PROCEDURE

RDSPS

GNFLTS

- FPSTAR
- ***PRFLTS***
  - FPSTAR
  - ***CKSPS***
  - ***GNBXY***
  - UNRELQ
    - FINTGRT
    - ***SUMMAT***

Figure A.1-9 GNFLTVC Call Tree

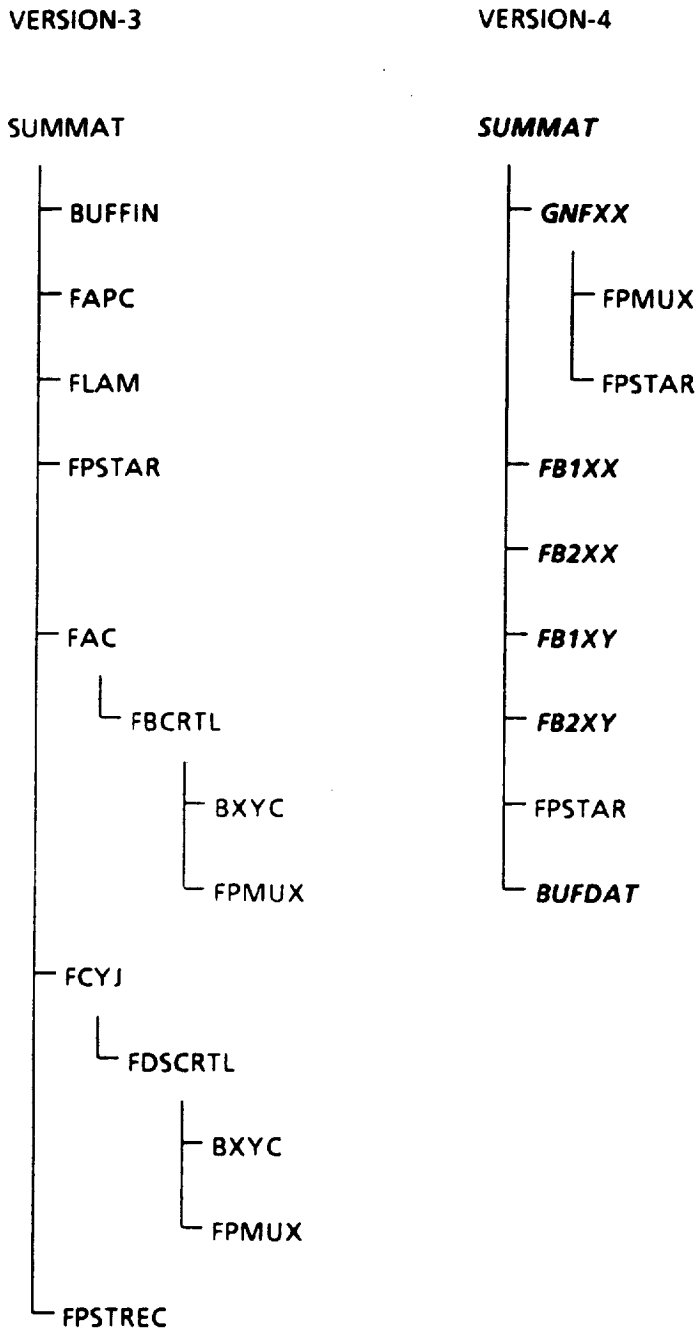


Figure A.1-10 SUMMAT Call Tree

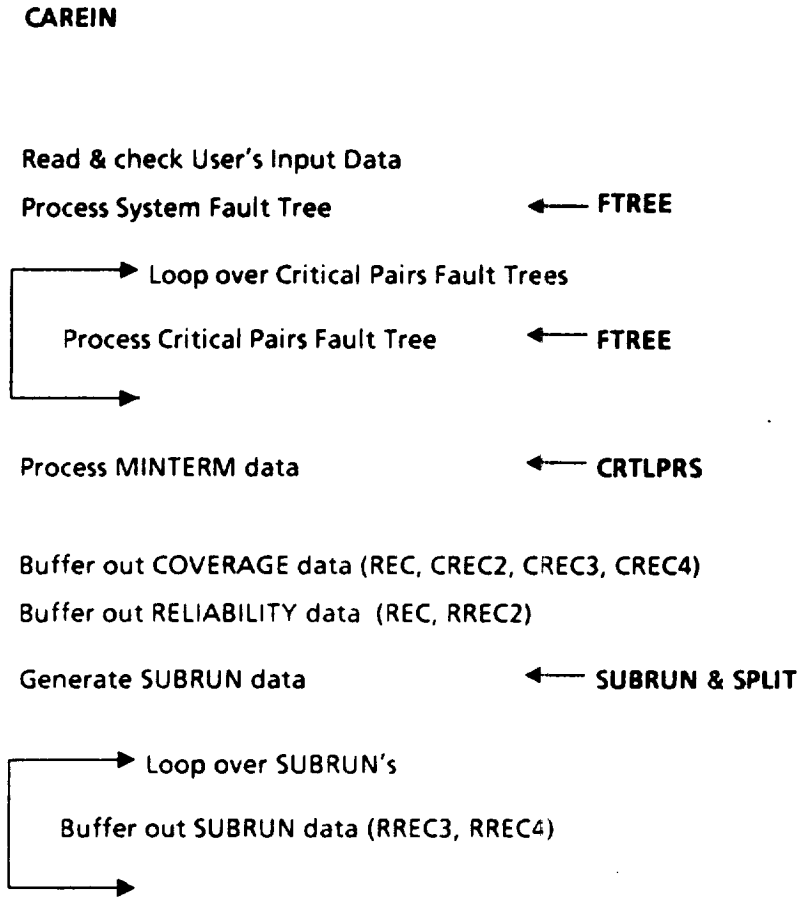


Figure A.2-1 CAREIN Design Sheet

CRTLPRS

Position I/O Units

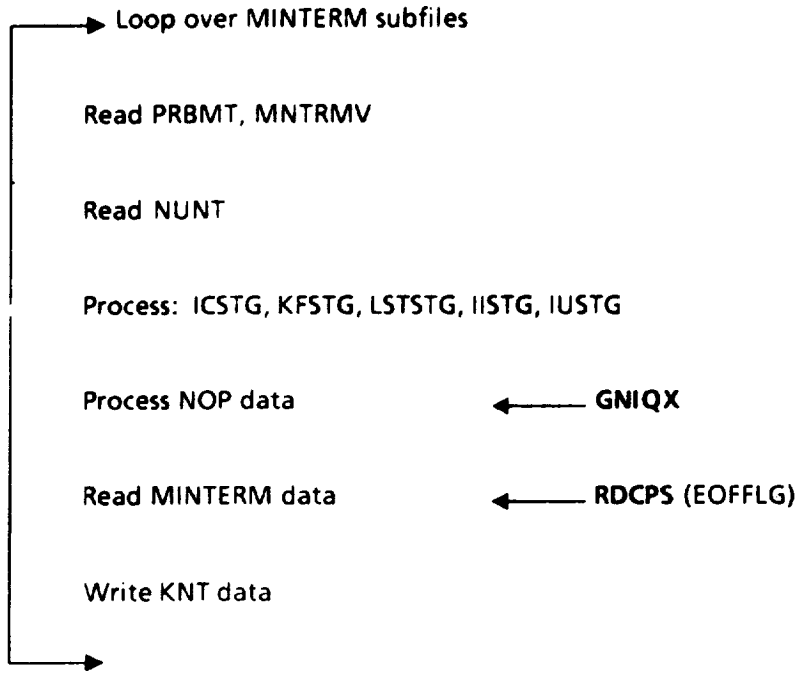


Figure A.2-2 CRTLPRS Design Sheet

GNIQX

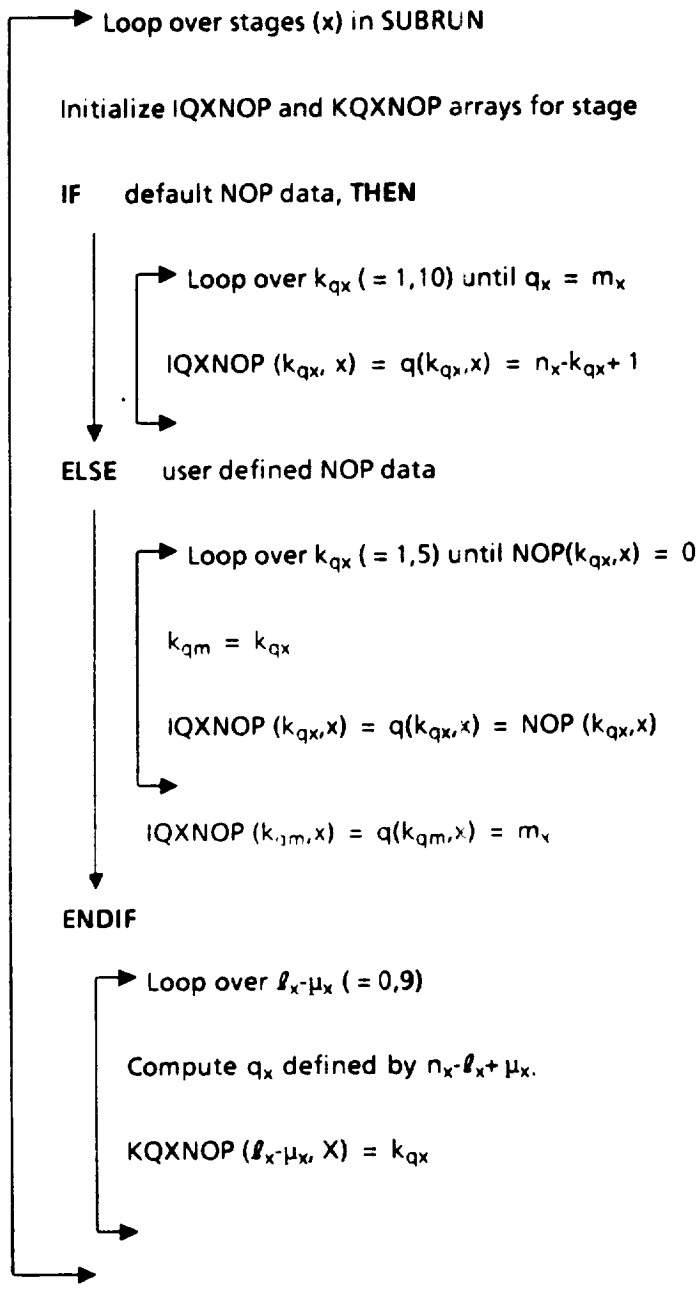


Figure A.2-3 GNIQX Design Sheet

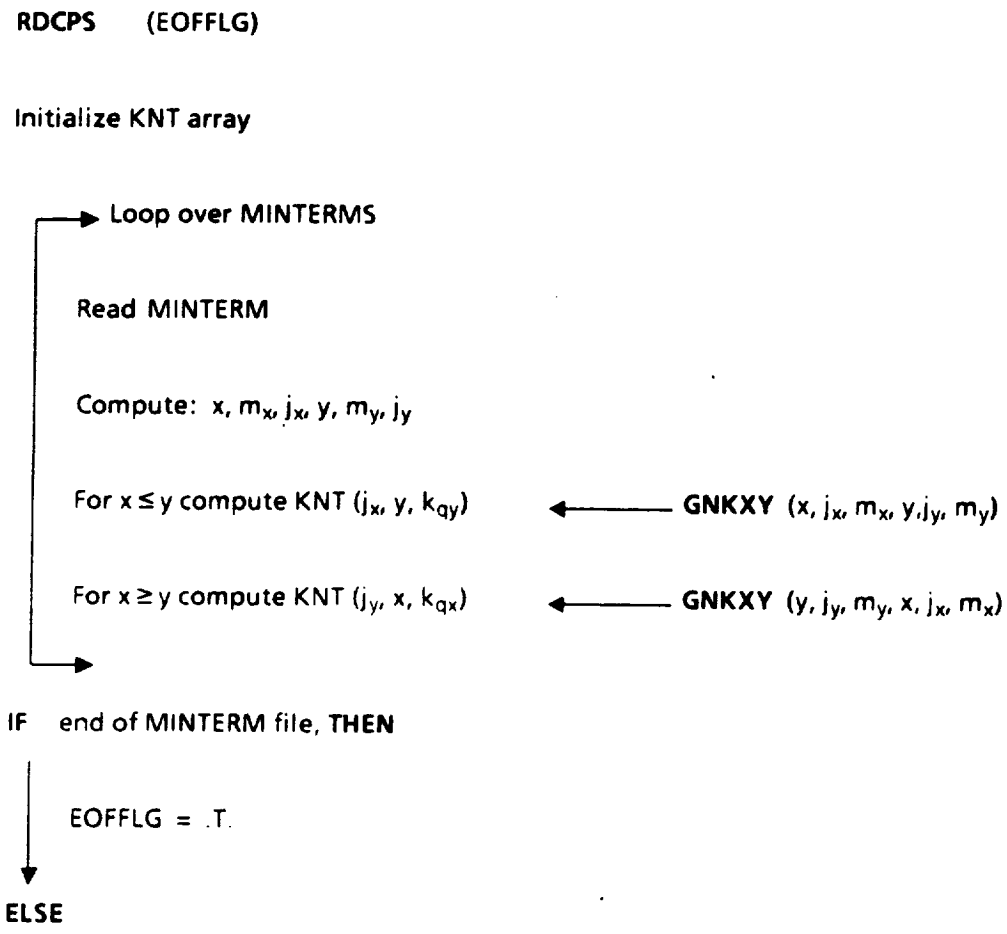


Figure A.2-4 RDCPS Design Sheet

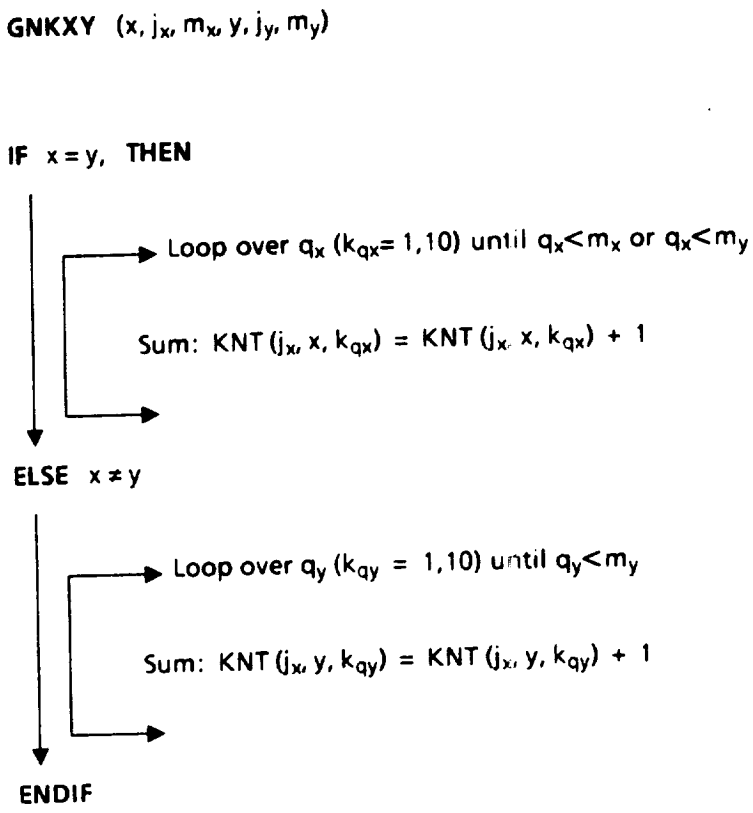


Figure A.2-5 GNKXY Design Sheet

MSNGFN

Obtain parameters for fault type

Initialize states and coefficient matrix

Initialize output coverage functions

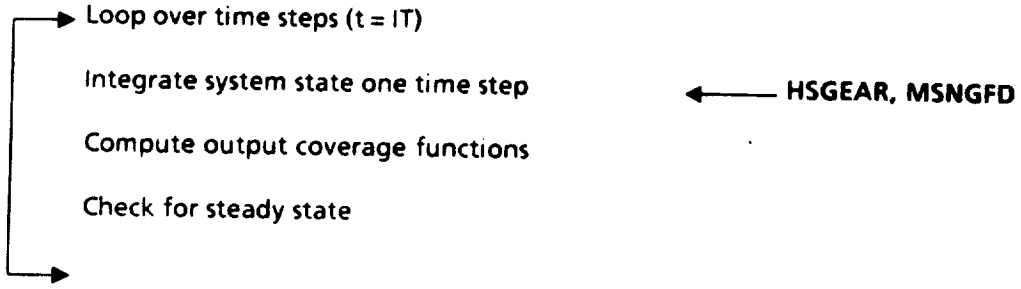


Figure A.2-6 MSNGFN Design Sheet

MSNGFD

Compute time derivatives of states

Figure A.2-7 MSNGFD Design Sheet

MSNGMT

Initialize moments for t = 0 (IT = 1)

Loop over time steps (IT = 2, ITSTPS)

Integrate weighted output coverage functions one reliability time step ← **HSGEAR, MSNGMD**

Store moments

Figure A.2-8 MSNGMT Design Sheet

MSNGMD (t)

Locate t in time array for output coverage functions

Compute weighted output coverage functions

Figure A.2-9 MSNGMD Design Sheet

MDBLFN

Obtain parameters for fault type

Initialize states and coefficient matrix

Initialize output coverage functions

Loop over time steps (t = IT)

Integrate system state one time step ← **HSGEAR, MDBLFD**

Compute output coverage functions

Check for steady state

Figure A.2-10 MDBLFN Design Sheet

MDBLFD

Compute time derivatives of states

Figure A.2-11 MDBLFD Design Sheet

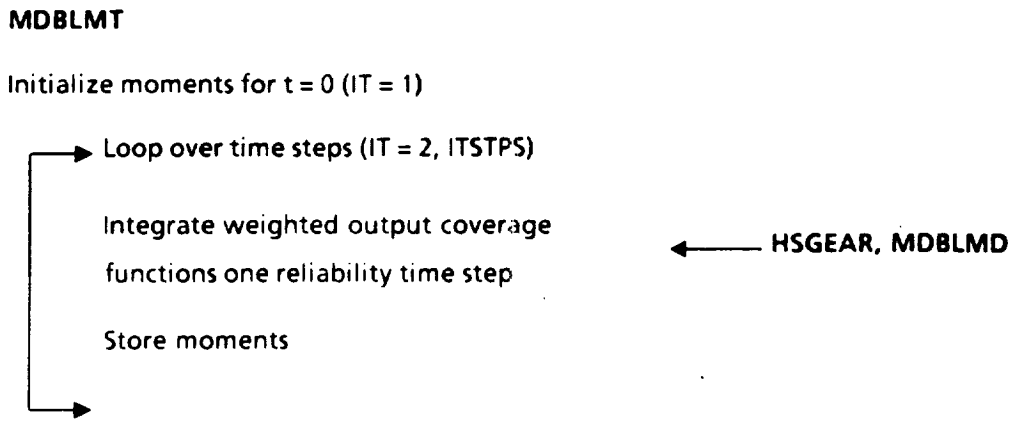


Figure A.2-12 MDBLMT Design Sheet

MDBLMD (t)

Locate t in time array for output coverage functions

Compute weighted output coverage functions

Figure A.2-13 MDBLMD Design Sheet

CARE3

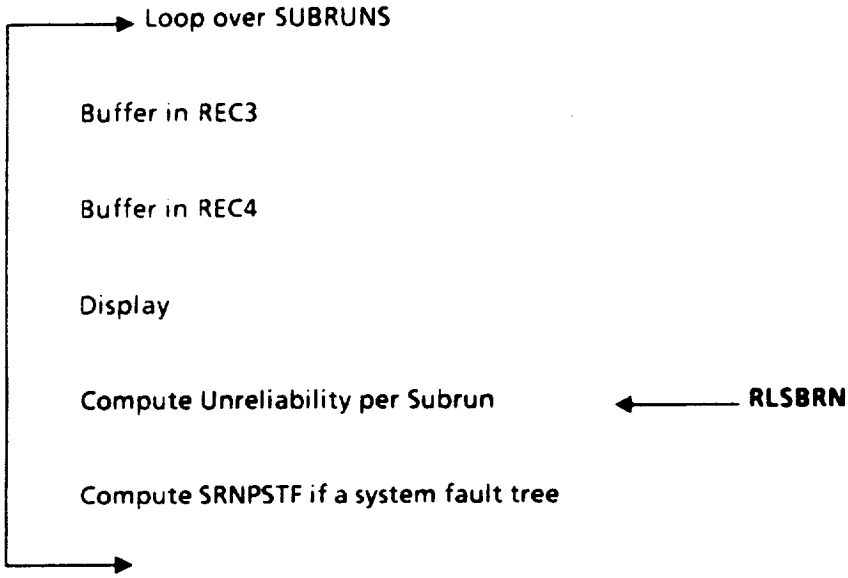
Buffer in CVRGAR array

Buffer in TITLE array

Buffer in REC1

Buffer in REC2

Compute KWT from system MINTERM file



Compute P* if a system fault tree

Write SUMMARY

Figure A.2-14 CARE3 Design Sheet

RLSBRN

Convert failure rates to correct time base

Create TRNSFC array

Compute non-$\underline{\ell}$ dependent functions ← **NFLTVDP**

IF Version 3 fault generation procedure, **THEN**

Generate fault vectors ← **GNFLTVC**

ELSE Version 4 fault generation procedure

Extract SUBRUN fault tree ← **RDSPS**

Generate fault vectors ← **GNFLTS**

ENDIF

Figure A.2-15 RLSBRN Design Sheet

NFLTVDP

Compute GFLD

Buffer in KNT data

IF Critical Pairs for SUBRUN, **THEN**

Process SUBRUN for which Critical Pairs are defined

ELSE

Process SUBRUN for which no Critical Pairs are defined

ENDIF

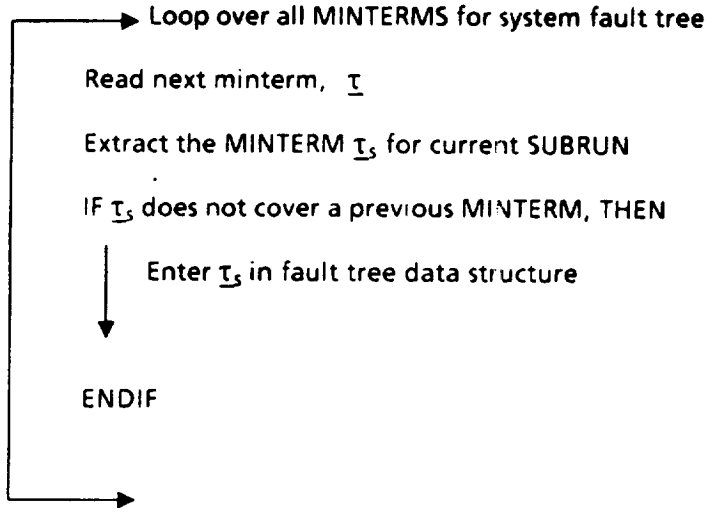
Generate NXX and NXY data ← **GNCPS**

Generate BXX and BXY data ← **GNBPS**

Figure A.2-16 NFLTVDP Design Sheet

RDSPS

Position I/O units



Note: The logic in RDSPS and the order of MINTERM's stored on the system minterm file assures that only a MINCUT set of minterms is stored for the SUBRUN fault tree.

Figure A.2-17 RDSPS Design Sheet

GNFLTS

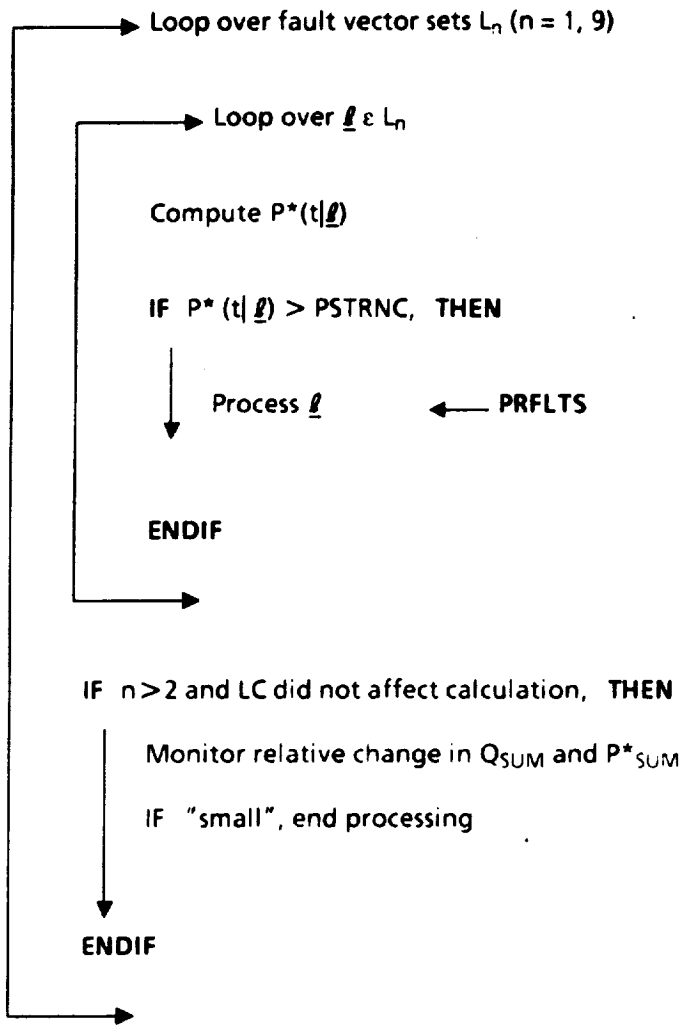


Figure A.2-18 GNFLTS Design Sheet

PRFLTS

IF $\underline{\ell} \neq 0$, check if $\underline{\ell}$ causes systems failure ← **CKSPS** ($I\underline{\ell}$)

Case: $\underline{\ell} = 0$

- Initialize display formats
- Compute Q(t|0) ← **UNRELQ**
- Display

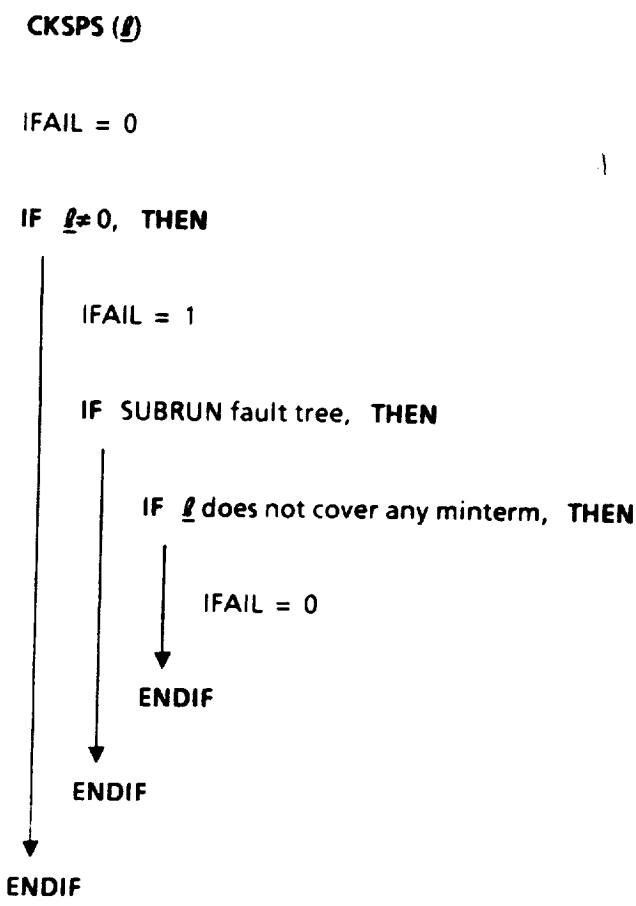
Case: $\underline{\ell} \neq 0$. $\underline{\ell}$ does not cause system failure

- Compute Q(t|$\underline{\ell}$) ← **UNRELQ**
- Display

Case: $\underline{\ell} \neq 0$, $\underline{\ell}$ causes system failures

- Compute P*(t,$\underline{\ell}$) ← **FPSTAR**
- Display

Figure A.2-19 PRFLTS Design Sheet



Note: If there is no user supplied system fault tree or if the extracted set of MINTERMS for a SUBRUN is empty, the SUBRUN fault tree is assumed to be an OR tree.

Figure A.2-20 CKSPS Design Sheet

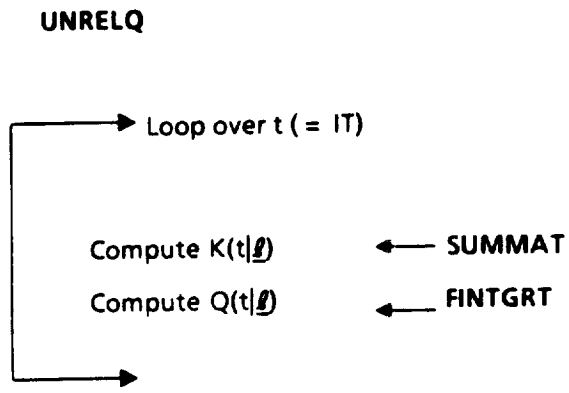


Figure A.2-21 UNRELQ Design Sheet

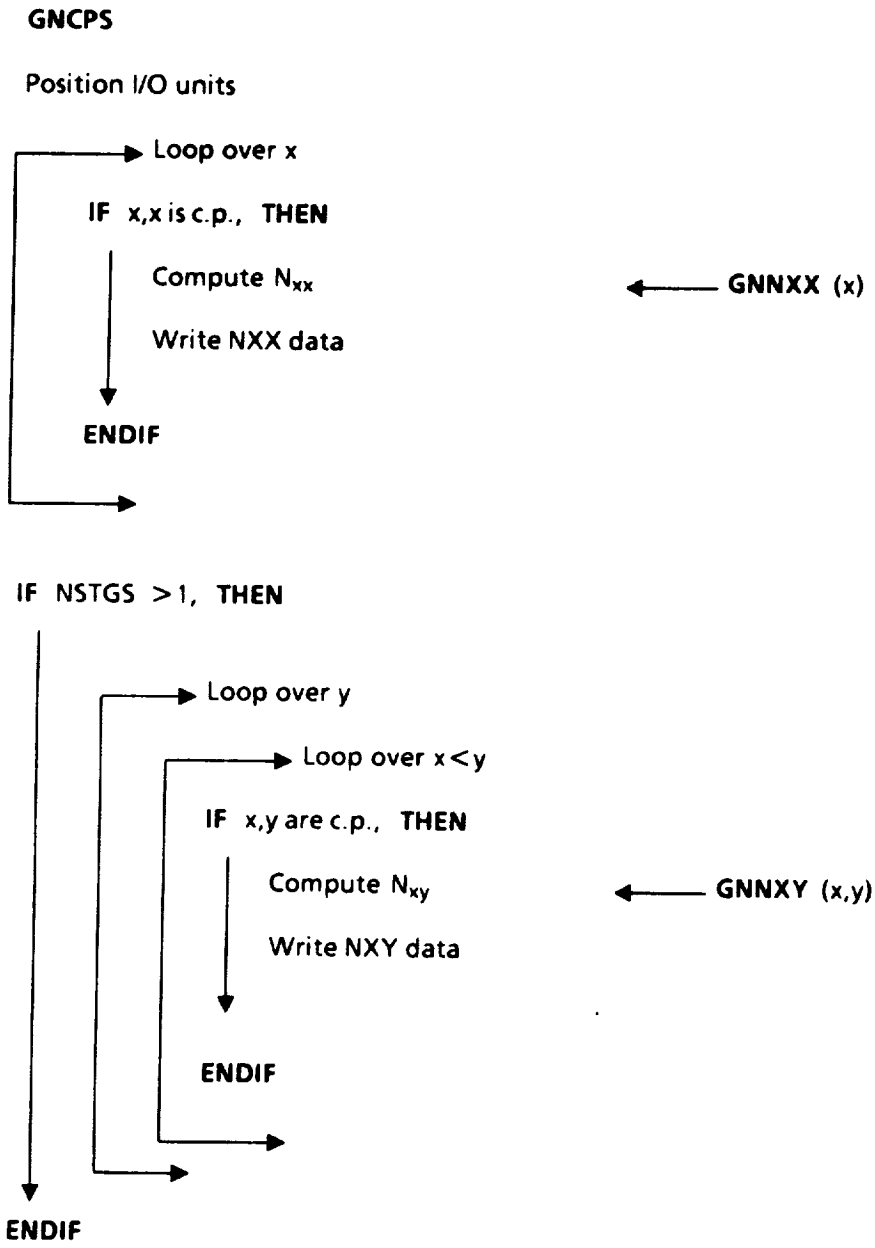


Figure A.2-22 GNCPS Design Sheet

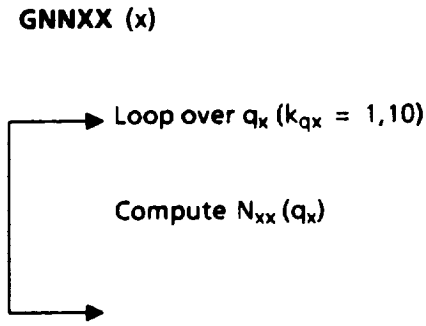


Figure A.2-23 GNNXX Design Sheet

GNNXY (x,y) [assume x<y]

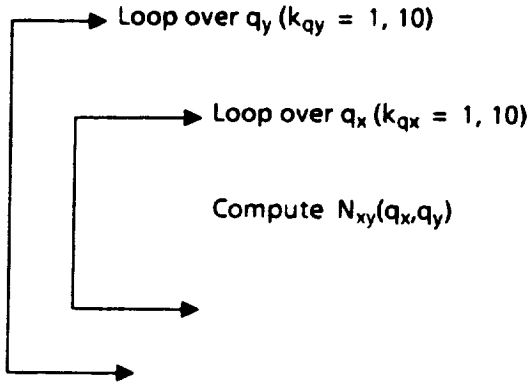


Figure A.2-24 GNNXY Design Sheet

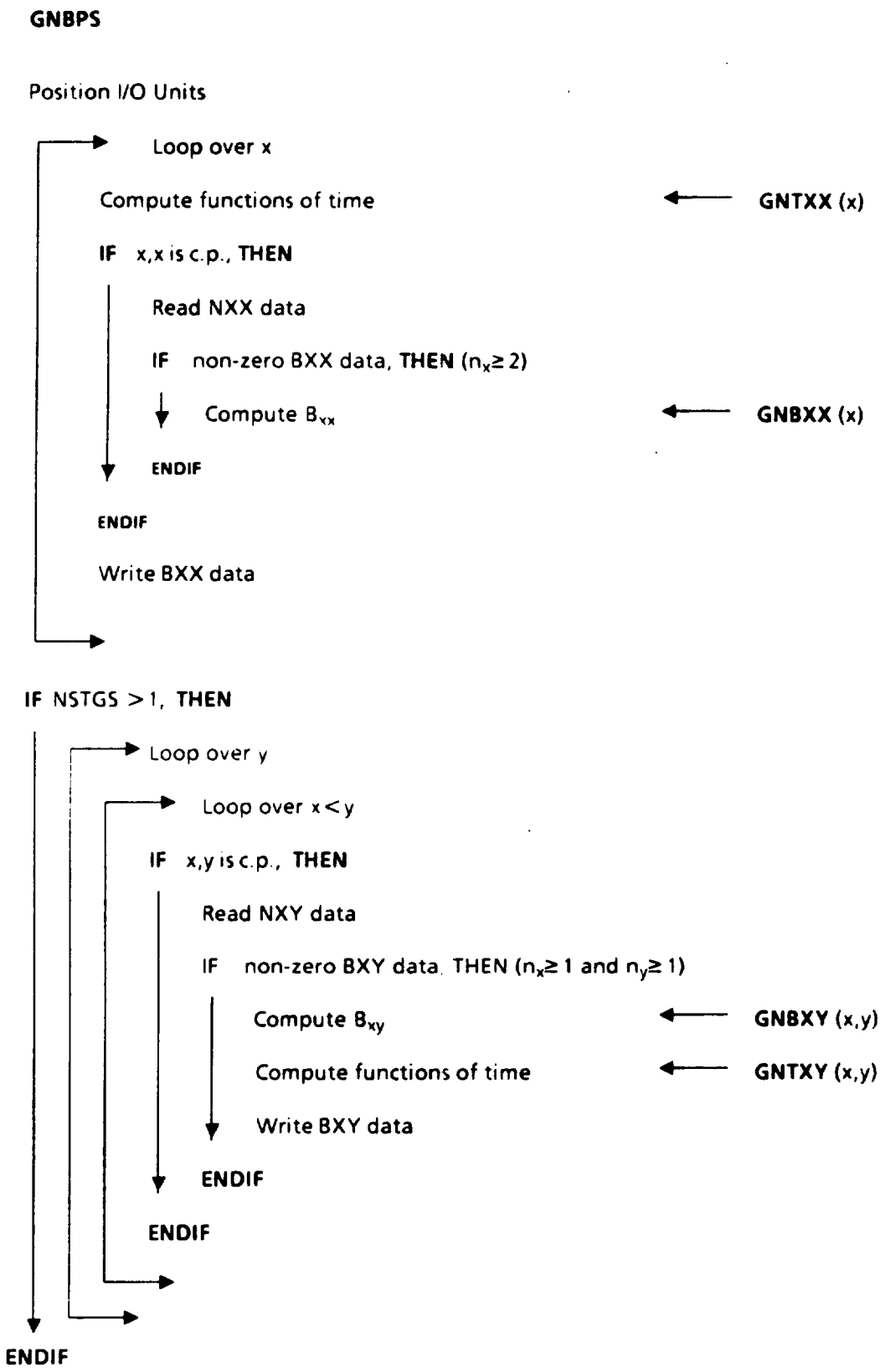


Figure A.2-25 GNBPS Design Sheet

GNBXX (x)

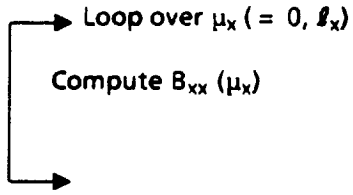


Figure A.2-26 GNBXX Design Sheet

GNB1XY (x,y) [Assume x<y]

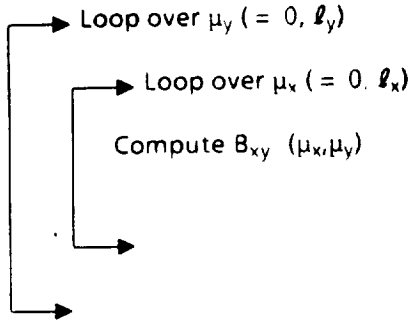


Figure A.2-27 GNBXY Design Sheet

GNTXX (x,y)

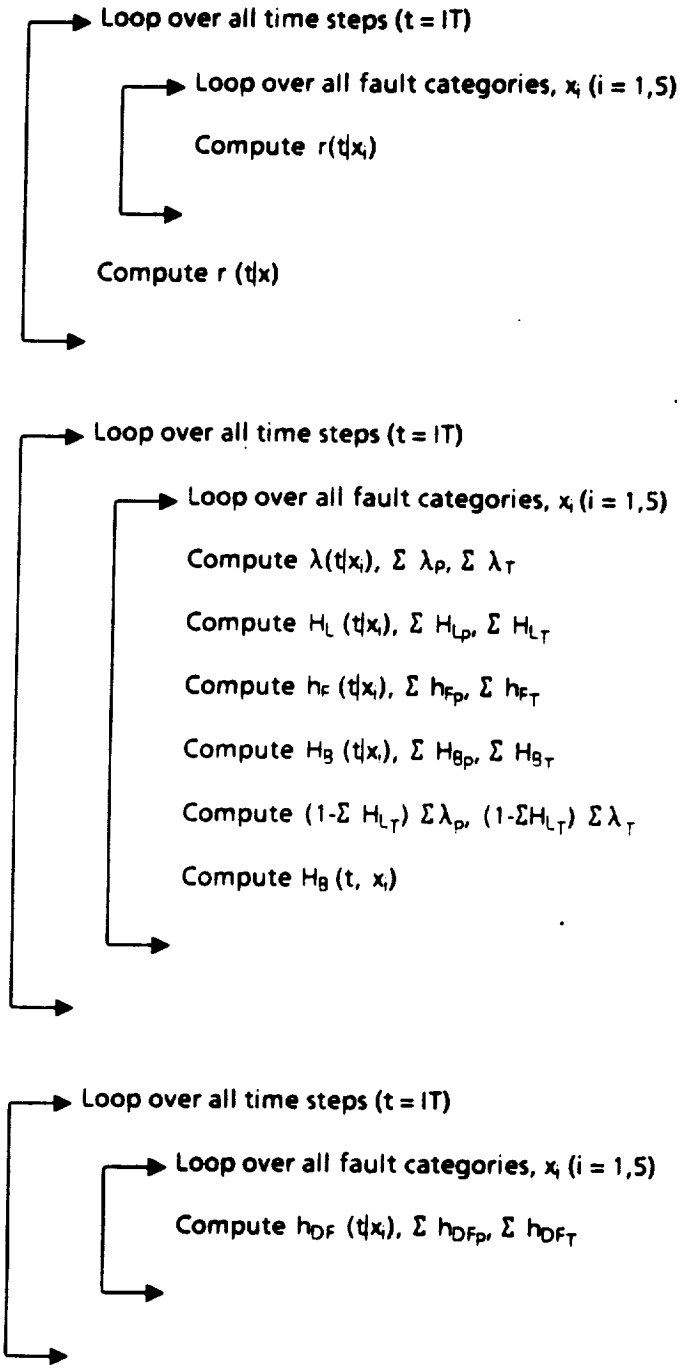


Figure A.2-28 GNTXX Design Sheet

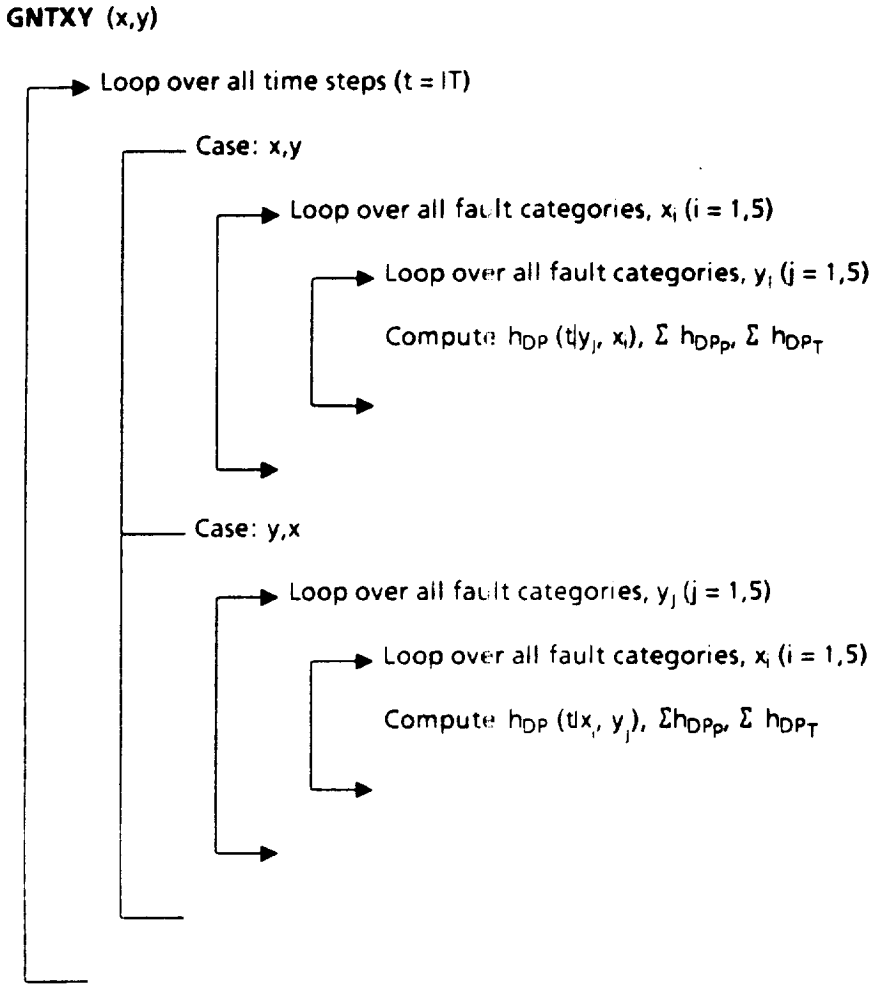


Figure A.2-29 GNTXY Design Sheet

SUMMAT (t|ℓ)

Position I/O units, Process IS, Initialize

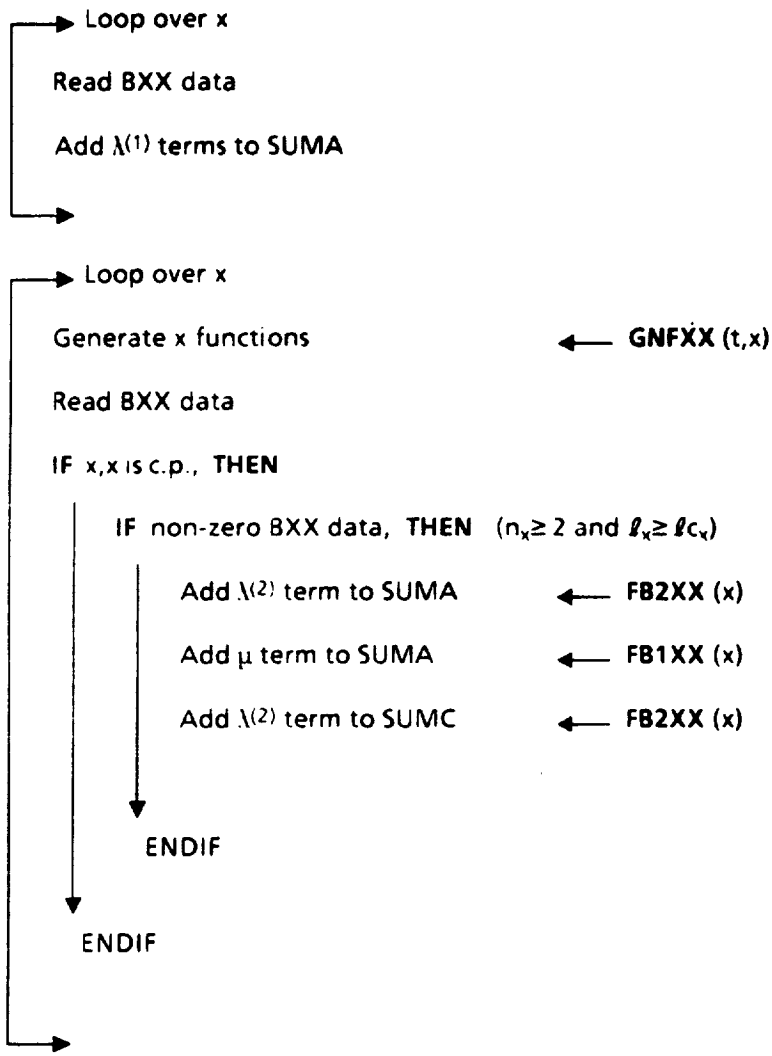


Figure A.2-30 SUMMAT Design Sheet

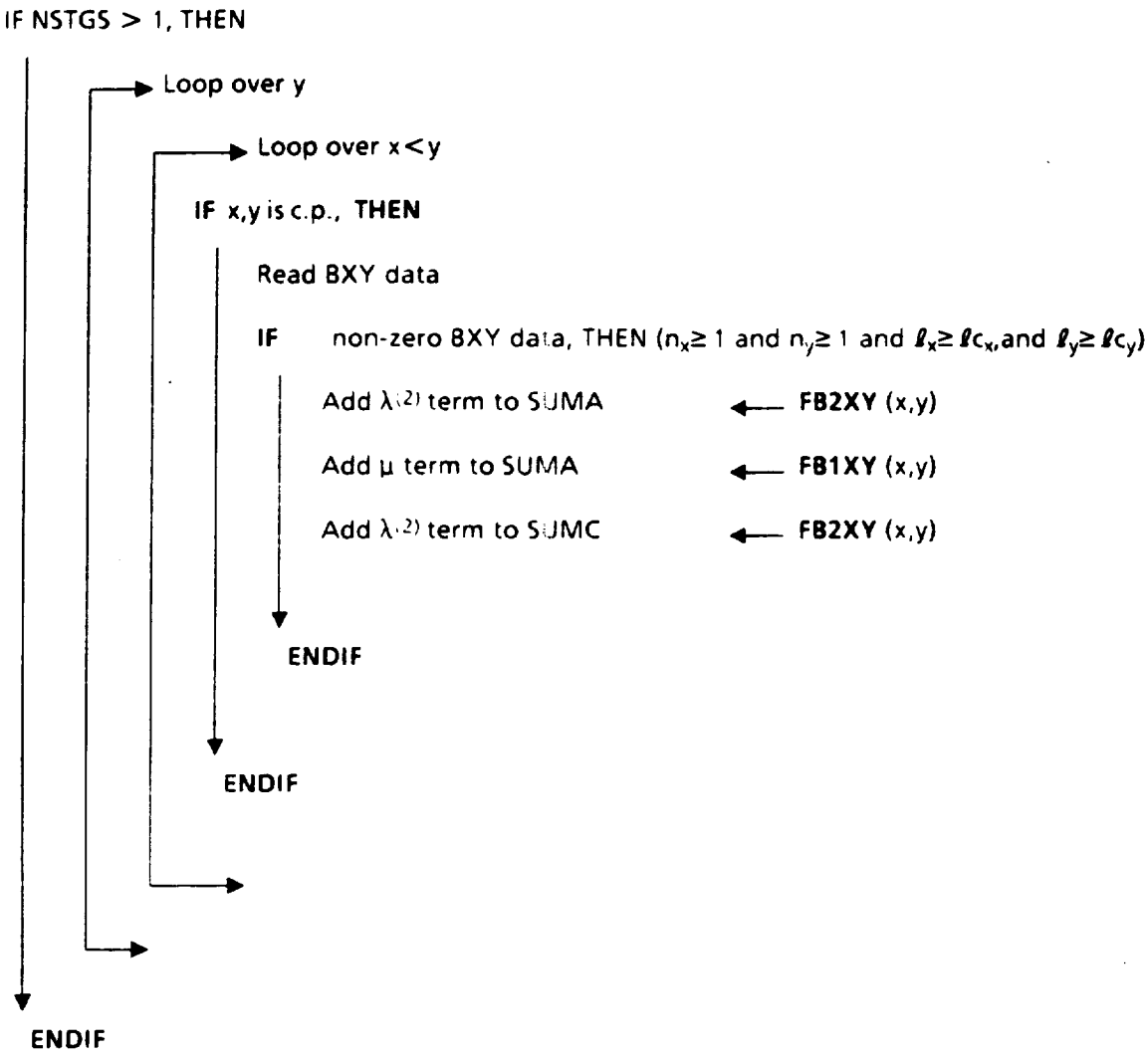


Figure A.2-30 SUMMAT Design Sheet (Continued)

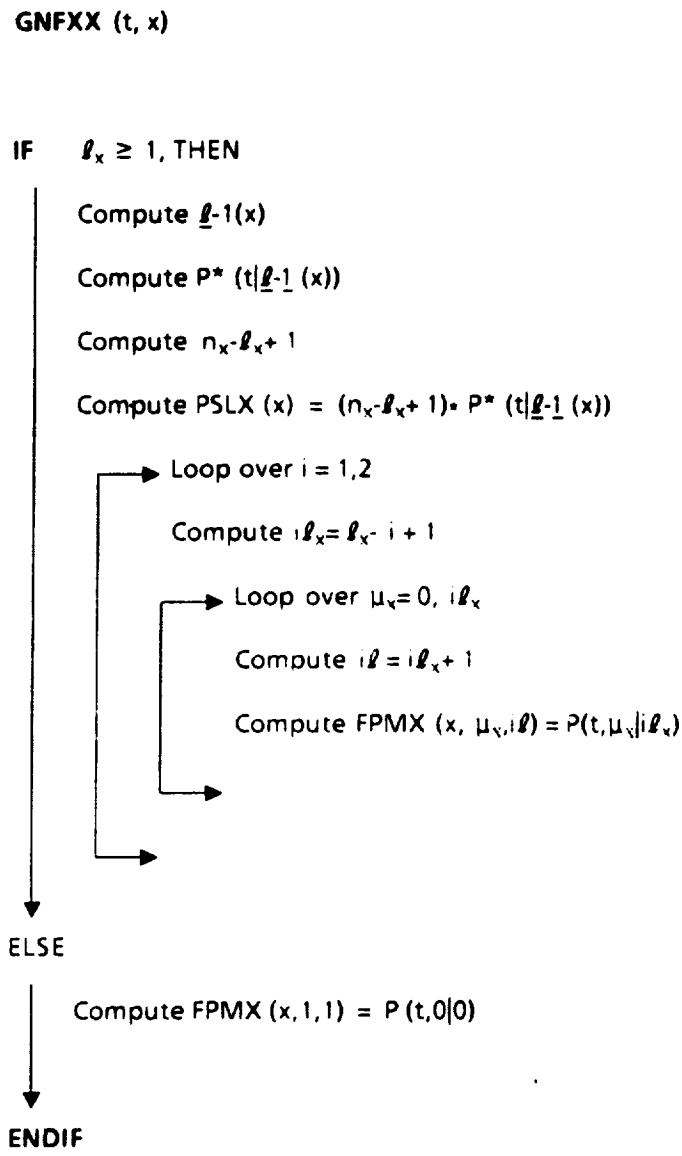


Figure A.2-31 GNFXX Design Sheet

FB1XX (x)

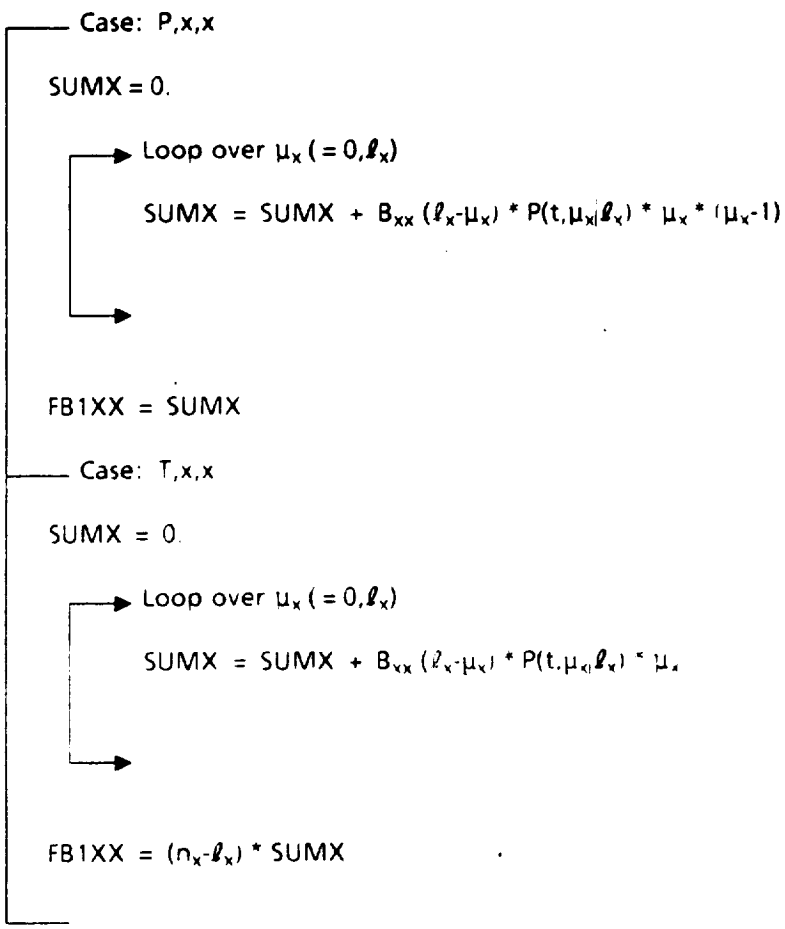


Figure A.2-32 FB1XX Design Sheet

FB1XY (x,y)

Case: P,x,y

SUMY = 0.

Loop over μ_y (= 0,ℓ_y)

SUMX = 0.

Loop over μ_x (= 0,ℓ_x)

SUMX = SUMX + B_{xy} (ℓ_x-μ_x, ℓ_y-μ_y) * P(t,μ_x|ℓ_x) * μ_x

SUMY = SUMY + SUMX * P(t,μ_y|ℓ_y) * μ_y

FB1XY = SUMY

Case: P,y,x

SUMX = 0.

Loop over μ_x (= 0,ℓ_x)

SUMY = 0.

Loop over μ_y (= 0,ℓ_y)

SUMY = SUMY + B_{xy} (ℓ_y-μ_y, ℓ_x-μ_x) * P(t,μ_y|ℓ_y) * μ_y

SUMX = SUMX + SUMY * P(t,μ_y|ℓ_y) * μ_y

FB1XY = SUMX

Figure A.2-33 FB1XY Design Sheet

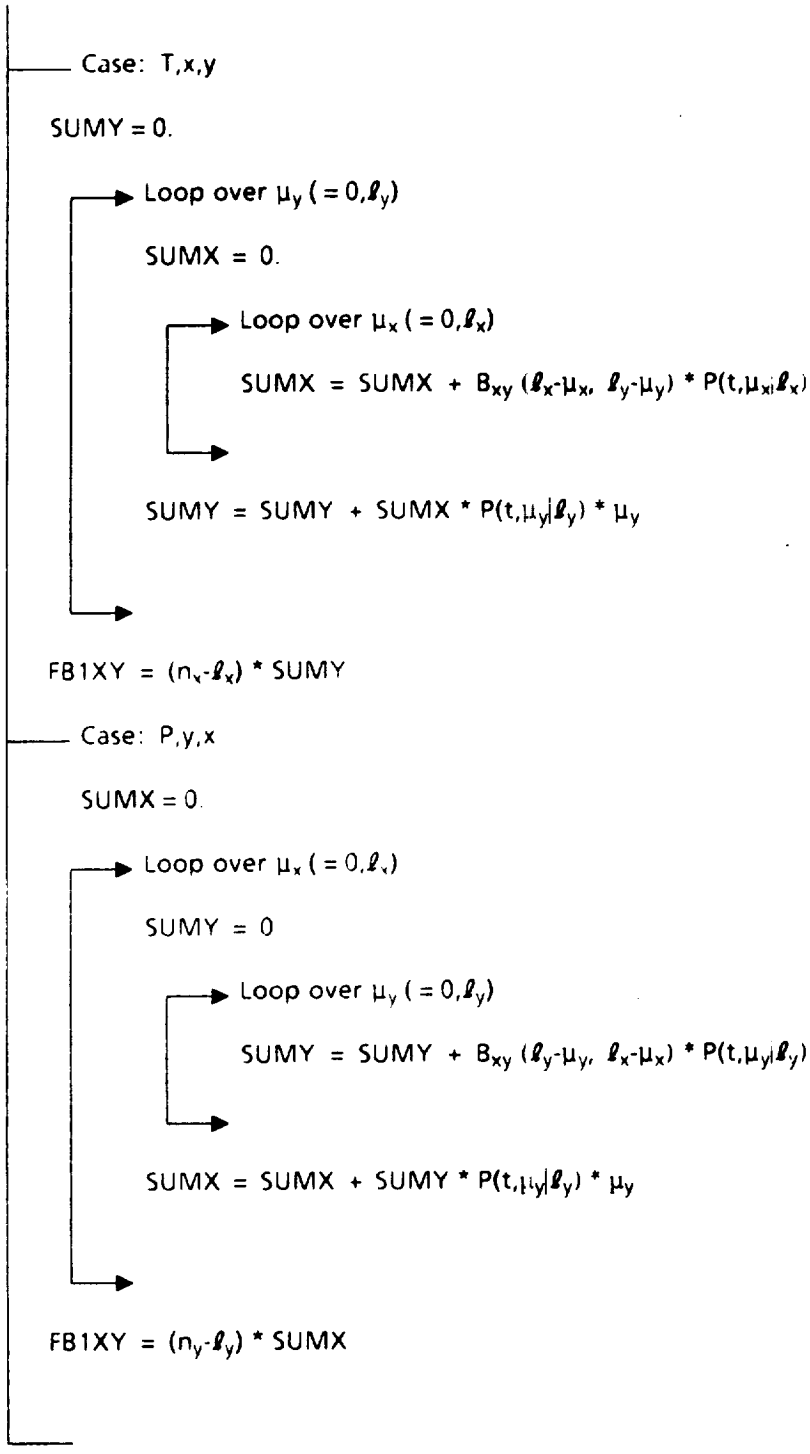


Figure A.2-33 FB1XY Design Sheet (Continued)

FB2XX (x)

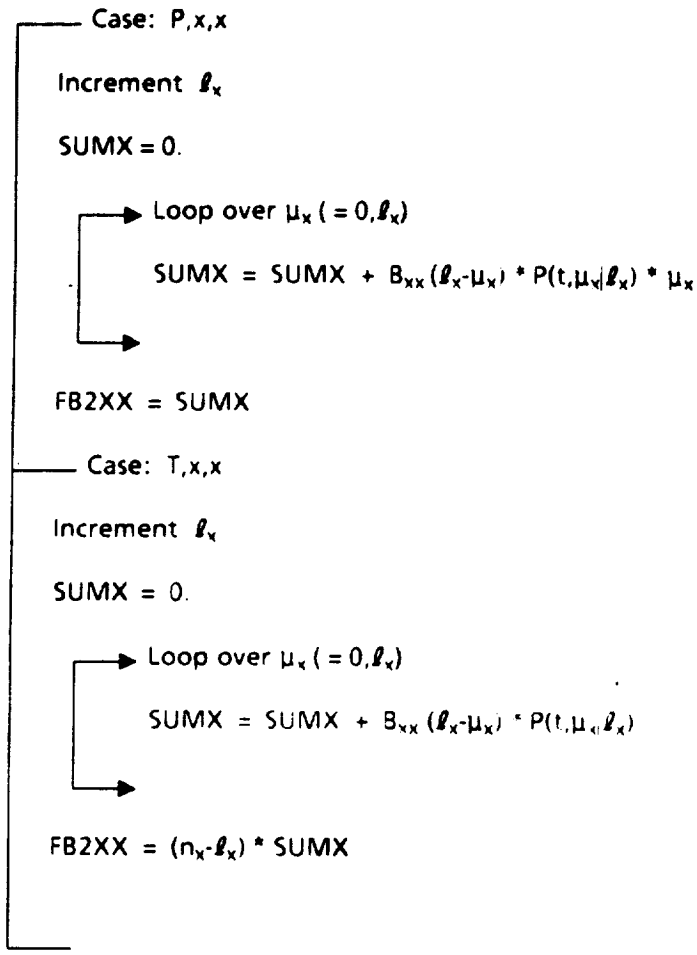


Figure A.2-34 FB2XX Design Sheet

FB2XY (x,y)

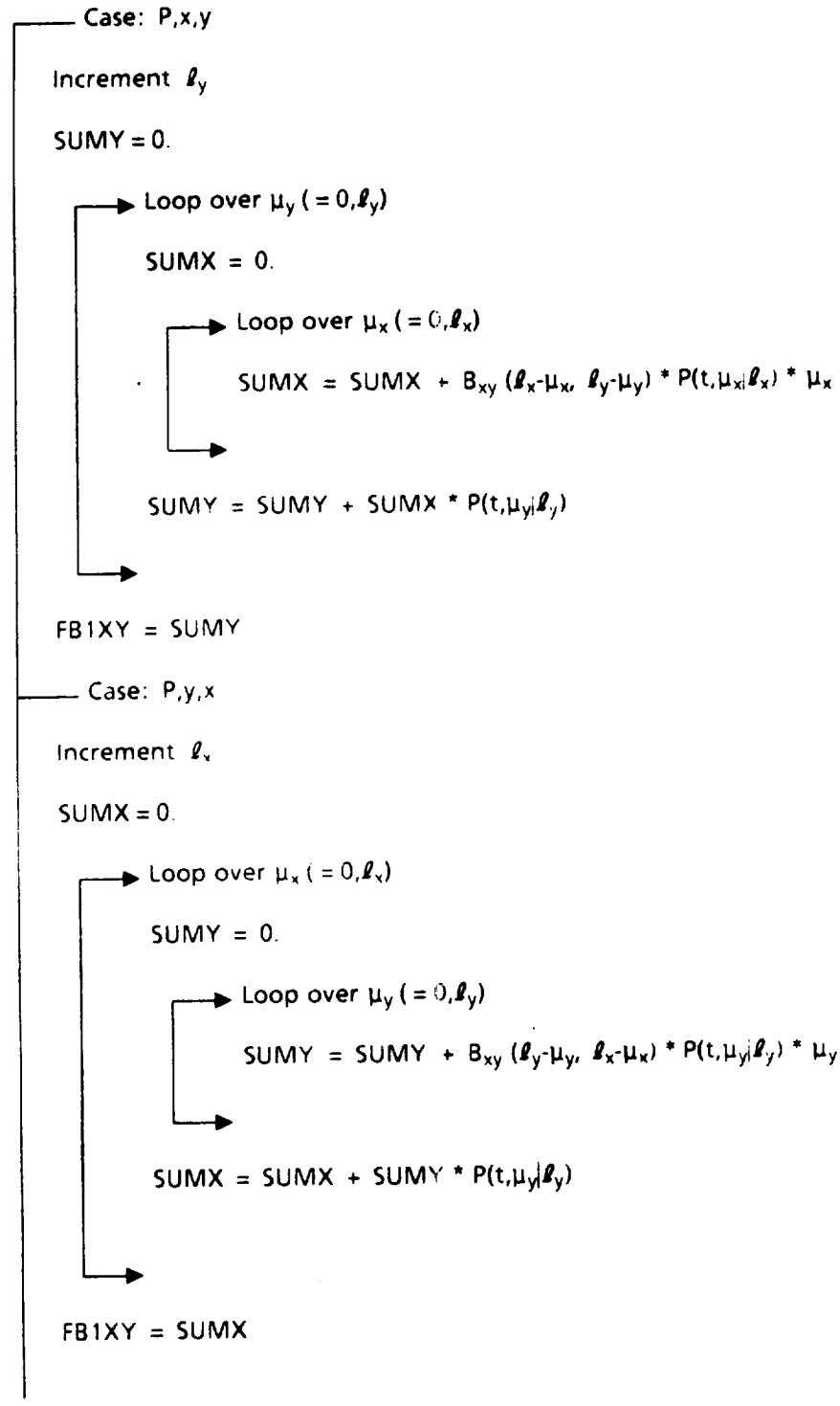


Figure A.2-35 FB2XY Design Sheet

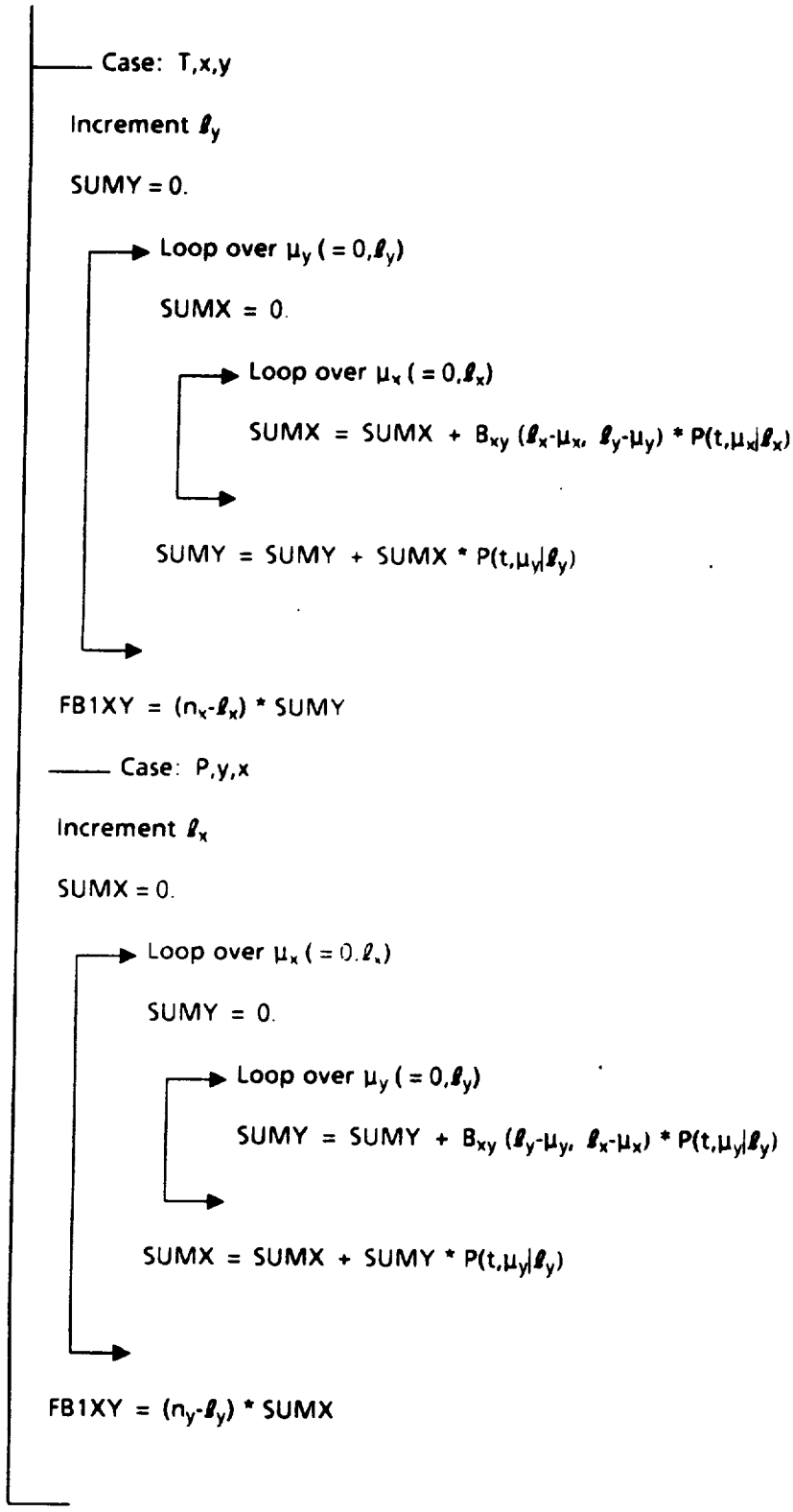


Figure A.2-35 FB2XY Design Sheet (Continued)

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-166122, Revised		

4. Title and Subtitle	5. Report Date
Correction, Improvement and Model Verification of CARE III, Version 3	December 1987
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
D. M. Rose, J. W. Manke, R. E. Altschul, and D. L. Nelson	
	10. Work Unit No. 505-34-43-05

9. Performing Organization Name and Address	11. Contract or Grant No.
Boeing Computer Services Company Energy Technology Applications Division Seattle, WA 98124	NAS1-16900
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

NASA Project Engineer: Salvatore J. Bavuso, NASA Langley Research Center, Hampton, VA 23665

Interim Report

Supersedes NASA CR-166122 dated April 1983.

16. Abstract

An independent verification of the CARE III mathematical model and computer code was conducted and reported in NASA Contractor Report 166096, "Review and Verification of CARE III Mathematical Model and Code: Interim Report." The study uncovered some implementation errors that were corrected and are reported in this document. The corrected CARE III program is called version 4. Thus the document, "Correction, Improvement and Model Verification of CARE III," version 3 was written in April 1984. It is being published now as it has been determined to contain a more accurate representation of CARE III than the published document that preceded it in April 1983. This edition supersedes NASA CR-166122 entitled, "Correction and Improvement of CARE III," version 3, April 1983.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Reliability modeling CARE III Fault coverage Fault models Fault-tolerant avionics	unclassified - unlimited Subject category 59

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
unclassified	unclassified	104	A06